Algorithmes de machine learning en assurance : solvabilité, textmining, anonymisation et transparence

Antoine Ly

École doctorale 532 : Mathématiques et STIC (MSTIC)

THÈSE

pour obtenir le grade de docteur délivré par

l'Université PARIS-EST Marne-la-Vallée (UPEM)

Spécialité doctorale "Mathématiques et leurs interactions"

présentée et soutenue publiquement par

Antoine LY

le 19 novembre 2019

Algorithmes de machine learning en assurance : solvabilité, textmining, anonymisation et transparence.

Entreprise d'accueil : **Milliman**
Directeur de thèse : **Romuald Elie (UPEM)**
Co-encadrant de thèse : **Arthur Charpentier (UQAM)**

Jury		
Hainaut,	Donatien	Rapporteur
Loisel,	Stéphane	Rapporteur
Rossi,	Fabrice	Président
Boumezoued,	Alexandre	Examinateur
Kratz,	Marie	Examinateur

UPEM
LABORATOIRE D'ANALYSE ET DE MATHÉMATIQUES APPLIQUÉES
UMR CNRS 8050, Université Paris-Est Marne la Vallée,
bâtiment Copernic 5 boulevard Descartes 77454, MARNE LA VALLÉE, France

RÉSUMÉ

En été 2013, le terme de "Big Data" fait son apparition et suscite un fort intérêt auprès des entreprises. Cette thèse étudie ainsi l'apport de ces méthodes aux sciences actuarielles. Elle aborde aussi bien les enjeux théoriques que pratiques sur des thématiques à fort potentiel comme l'*Optical Character Recognition* (OCR), l'analyse de texte, l'anonymisation des données ou encore l'interprétabilité des modèles. Commençant par l'application des méthodes du machine learning dans le calcul du capital économique, nous tentons ensuite de mieux illustrer la frontrière qui peut exister entre l'apprentissage automatique et la statistique. Mettant ainsi en avant certains avantages et différentes techniques, nous étudions alors l'application des réseaux de neurones profonds dans l'analyse optique de documents et de texte, une fois extrait. L'utilisation de méthodes complexes et la mise en application du Réglement Général sur la Protection des Données (RGPD) en 2018 nous a amené à étudier les potentiels impacts sur les modèles tarifaires. En appliquant ainsi des méthodes d'anonymisation sur des modèles de calcul de prime pure en assurance non-vie, nous avons exploré différentes approches de généralisation basées sur l'apprentissage non-supervisé. Enfin, la réglementation imposant également des critères en terme d'explication des modèles, nous concluons par une étude générale des méthodes qui permettent aujourd'hui de mieux comprendre les méthodes complexes telles que les réseaux de neurones.

Mots clés : Apprentissage Statistique, Assurance, Reconnaissance Optique de Caractères (OCR), Anonymisation, Analyse de texte, Interprétabilité, Cas d'application.

ABSTRACT

In summer 2013, the term "Big Data" appeared and attracted a lot of interest from companies. This thesis examines the contribution of these methods to actuarial science. It addresses both theoretical and practical issues on high-potential themes such as *Optical Character Recognition* (OCR), text analysis, data anonymization and model interpretability. Starting with the application of machine learning methods in the calculation of economic capital, we then try to better illustrate the boundary that may exist between automatic learning and statistics. Highlighting certain advantages and different techniques, we then study the application of deep neural networks in the optical analysis of documents and text, once extracted. The use of complex methods and the implementation of the General Data Protection Regulation (GDPR) in 2018 led us to study its potential impacts on pricing models. By applying anonymization methods to pure premium calculation models in non-life insurance, we explored different generalization approaches based on unsupervised learning. Finally, as regulations also impose criteria in terms of model explanation, we conclude with a general study of methods that now allow a better understanding of complex methods such as neural networks.

Key words: Machine Learning, Insurance, Optical Character Recognition (OCR), Anonymization, Text Analysis, Interpretability, Use Cases.

A mes parents.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont permis de m'épanouir pendant ces quatre années de thèse. Je remercie ainsi l'entreprise Milliman et ses directeurs Eric Serant et Jérôme Nebout de m'avoir accueilli, fait confiance tout au long de ces années et fait découvrir tous ces sujets qui me passionnent aujourd'hui. Je remercie Fabien Conneau, mon premier tuteur au sein de Milliman pour m'avoir transféré son goût pour la recherche appliquée. Je remercie également Laurent Devineau, intialement mon directeur de thèse en entreprise, pour m'avoir inspiré tant d'un point de vue humain qu'intellectuel. Je remercie l'équipe R&D de Milliman et particulièrement Alexandre Boumezoued, Pierre-Edouard Arrouy et Paul Bonnefoy pour les différents échanges que nous avons pu avoir. Je remercie Isaac Haik, pour les travaux que nous avons menés ensemble. Je remercie enfin Rémi Bellina, Thomas Poinsignon, Dimitri Delcaillau de l'équipe analytique Milliman pour les différentes collaborations de ces dernières années.

Je tiens également à remercier tous mes amis et mes proches pour le soutien et la joie qu'ils m'apportent au quotidien. Je remercie mes amis et plus particulièrement mes témoins de mariage Antoine Guillot et Loic Michel pour l'inspiration qu'ils me donnent. Je remercie également Thibault Laugel pour sa bonne humeur, son support et les échanges sur l'interprétabilité des modèles. Je remercie également Xavier Dupré pour avoir su également me motiver dans les moments difficiles et tous les échanges enrichissants que nous avons pu avoir.

Je remercie Audrey, ma fiancée, pour son aide dans la relecture et d'avoir toujours su me remonter le moral dans les moments difficiles. Elle est un facteur d'aboutissement de ces travaux.

Je remercie également mes soeurs et mes parents pour leur confiance et l'amour qu'ils me portent.

Je remercie mes directeurs de thèse Romuald Elie et Arthur Charpentier pour m'avoir soutenu et convaincu d'aller au bout de ces travaux. Je ne saurais être assez reconnaissant pour leur soutien et leur amitié.

Enfin, je remercie les rapporteurs et le jury pour l'attention apportée à cette thèse et les différents retours partagés.

CONTENTS

ACRONYMES

ACPR Autorité de Contrôle Prudentiel et de Résolution

ALE Accumulated Local Effects

ANR Actifs Nets Réévalués

CART Classification and Regression Tree

CNN Réseaux de Neurones Convolutionnels

EIOPA European Insurance and Occupationnal Pensions Authority

GB Gradient Boosting

GLM Generalized Linear Models

HMM Hidden Markov Model

IBM Interpretabilité Basée sur le Modèle

NAV Net Asset Value

NER Name Entity Recognition

NLP Natural Language Processing

OCR Optical Character Recognition

PDP Partial Dependence Plot

RF Random Forest

RGPD Règlement Général sur la Protection des Données

RNN Réseaux de Neurones Récurrents

SDS Simulations dans les Simulations

VIF Value In Force

1

INTRODUCTION

En été 2013, le terme de "Big Data" [1] fait son apparition et suscite un fort intérêt auprès des entreprises [2]. L'utilisation de l'information numérique afin d'enrichir la compréhension de phénomènes financiers, marketings, biométriques ou encore comportementaux alimente alors tous les fantasmes. Dès lors, les industries découvrent et accumulent de nouvelles données dont l'acquisition est facilitée par des systèmes d'information distribuée[3] et le coût de stockage de plus en plus faible. L'analyse et l'exploitation de ces dernières s'accompagnent très rapidement d'algorithmes dont les origines se partagent entre les sciences informatiques et statistiques [Charpentier et al. 2019].

L'assurance fondée sur la modélisation du risque commence alors à s'interroger sur l'utilisation de nouveaux modèles : le *machine learning*[4]. Ces derniers performent alors sur des données de tout type (structurées ou non) et de différentes volumétries[5]. C'est donc dans ce contexte que je me suis interrogé sur le champ des possibles de l'utilisation de l'apprentissage statistique en assurance. Ces questions passent nécessairement par la compréhension de ces modèles atypiques pour le secteur industriel de l'assurance mais également de leurs limites. Cette thèse décline alors en différents chapitres les applications et apports possibles des algorithmes d'apprentissage statistique à différents secteurs de l'assurance. Chaque partie reflète un sujet de recherche appliquée qui met en avant la transposition ou l'extension de la recherche académique à des problématiques assurantielles. Cette thèse a par ailleurs été réalisée en entreprise (Milliman). A cet effet, chaque sujet abordé répond potentiellement à un besoin exprimé par une compagnie d'assurance. Les approches *machine learning* abordées dans ce manuscrit sont donc parfois critiquées sous un regard actuariel et opérationnel.

Mes travaux commencent par l'étude de l'apprentissage statistique et son potentiel apport dans la détermination d'un capital économique. En effet, depuis les différentes crises économiques et financières de cette dernière décennie, les institutions de réglementation

1 Données massives ou mégadonnées en français, le terme anglais est néanmoins majoritairement utilisé
2 D'après google.trends sur le mot clé "Big Data"
3 Le framework Hadoop est rendu public en 2008 et Spark en 2012
4 Apprentissage machine ou statistique en français
5 https://paperswithcode.com/

imposent aux compagnies d'assurance de se couvrir contre les risques futurs. Ainsi, la réglementation prudentielle Solvabilité II devant s'appliquant depuis janvier 2016 oblige les compagnies d'assurance à posséder suffisamment de fonds propres pour être capables d'empêcher toute ruine économique à un an avec une probabilité de 99.5%. D'autre part, l'apparition des objets connectés, des nouvelles données qu'ils engendrent ainsi que la multiplication des souscriptions d'assurance en ligne obligent les sciences actuarielles à repenser leurs modèles et à élargir leurs champs d'étude. Cette évolution semble nécessaire afin de tirer profit des nouvelles données et de s'adapter à un environnement de plus en plus concurrentiel. Si l'apprentissage statistique, encore peu utilisé en assurance en 2015, ouvre alors un nouveau domaine de recherche actuariel aussi bien dans le cadre d'étude des comportements des assurés que dans le calcul de capital économique.

En effet, la détermination de ce capital est parfois délicate. Une méthode envisageable est l'approche dite des Simulations dans les Simulations (SdS). Il s'agit par ce modèle, connaissant exactement les facteurs de risque d'une compagnie d'assurance de simuler, souvent sous deux probabilités différentes (monde réel puis risque neutre), de nombreux scénarios économiques à l'aide de méthodes de Monte Carlo. On suppose couvrir tous les scénarios économiques pouvant survenir dans un an. Ces scénarios étant directement liés à la valeur des passifs d'une entreprise on peut ajuster les fonds propres pour se couvrir contre les scénarios les plus risqués. Mais cette méthode de référence est coûteuse en temps de calcul. La complexité de simuler la valeur des passifs tient au fait qu'en assurance il existe des interactions complexes entre passifs et actifs. Une formule fermée de valorisation est souvent impossible en assurance pour estimer la valeur d'un passif à un horizon futur. De plus, les méthodes simulatoires comme le SdS nécessitent des temps de calcul très importants (ces temps peuvent parfois atteindre la semaine de calcul). C'est pourquoi des méthodes alternatives sont apparues comme les *Replicating Portfolios* [Devineau and Chauvigny 2010]. Cette méthode consiste par exemple à construire un portefeuille d'actifs financiers fictifs dont la valeur est proche de celle du passif de la compagnie d'assurance. Son avantage est de permettre, à l'issue de chaque scénario monde réel de première période, une évaluation approchée, rapide et simple, de la valeur des passifs. Cependant, ces alternatives sont encore peu satisfaisantes et amènent à s'orienter vers de nouvelles méthodes afin de réduire au maximum le biais des modèles alternatifs au SdS [Gramacy and Ludkovski 2014]. Nous avons donc étudié le potentiel de nouvelles méthodes comme les réseaux de neurones [Ly 2015]. Cette compréhension s'est déclinée par l'exercice d'implémentation d'un module en python que la formalisation du modèle qui permet de guider les développements.

Les résultats obtenus sont discutables. Si le modèle permet de gagner en précision, cette dernière est à relativiser au regard de la complexité du modèle introduit. La décomposition des facteurs d'erreur décrite dans ce chapitre nous permet de mieux comprendre les résultats. La complexité du calibrage réside dans la capacité à obtenir une variable

Table 1.: Aperçu des résultats du chapitre 2 selon les points de contrôle fournis par les différents modèles

Points de contrôle	OLS	SW	NNET monomes	NNET tous les FdR
246 561 359	283 741 287	255 705 222	327 190 351	302 801 413
200 188 938	210 284 748	206 714 739	267 091 632	257 907 865
357 259 026	318 905 212	316 809 525	383 265 312	326 452 974
208 330 461	225 853 837	187 020 409	258 871 509	239 504 919
319 892 503	336 927 535	322 722 084	395 569 942	371 539 936
1 417 153 023	1 428 590 425	1 425 842 221	1 408 550 811	1 391 869 482
824 628 324	850 039 770	840 773 722	833 074 421	833 730 385
119 869 630	123 683 122	114 800 756	129 750 661	137 700 854
-48 170 776	-51 399 548	-65 438 948	11 581 574	-64 510 905

cible peu bruitée.

Cette première étude a cependant permis de mettre en lumière le fonctionnement des librairies open source comme *Keras* ou *Tensorflow* -très populaires de nos jours- publiées sur python peu de temps après la réalisation de notre étude en 2015 [6]. Si la théorie des réseaux de neurones et notamment le théorème d'approximation universel et son corollaire (que nous démontrons), permet de justifier l'approche mathématique des réseaux de neurones dans l'approximation de fonctions continues, elle ne donne cependant pas d'indication quant à la quantité de neurones nécessaires à utiliser.

Une approche empirique dans l'architecture des réseaux de neurones est donc nécessaire. Même si la littérature présente des résultats séduisants dans l'utilisation de ces modèles, leur étude nous a amené à se questionner sur les limites de l'utilisation des algorithmes d'apprentissage statistique dans un cadre probabiliste comme celui des Simulations dans les Simulations. En outre, nos premiers travaux nous permettent de redéfinir la frontière entre les méthodes usuelles statistiques et probabilistes et celle de l'apprentissage statistique.

Dans ce mouvement collectif en 2015 où le *deeplearning*[7] semble sur-performer de nombreux états de l'art, il semblait pertinent d'être en mesure de comprendre pourquoi et surtout dans quelles conditions cette performance pouvait être prometteuse dans des cas d'application concrets de l'industrie de l'assurance. Comme les actuaires se doivent de contrôler les modèles qu'ils implémentent, nous nous sommes tout d'abord intéressés aux éventuels points communs et différences fondamentales entre le *machine learning*

6 Nos travaux ont été réalisés en python. De nombreuses librairies existaient déjà sous Linux comme *Theano* ou encore *Torch* mais étaient moins accessibles qu'en 2016 sous python

7 Apprentissage profond en français faisant souvent référence à l'utilisation de réseaux de neurones complexes.

et les méthodes traditionnelles utilisées en assurance à savoir le plus souvent des méthodes économétriques. L'article [Charpentier et al. 2019] auquel nous avons contribué justifie alors les motivations d'une telle utilisation et tente de mieux comprendre en quoi l'apprentissage statistique permet une extension -ou du moins propose un nouveau regard- des méthodes statistiques ou économétriques traditionnellement utilisées par les actuaires.

L'économétrie et l'apprentissage machine semblent en effet avoir une finalité en commun : construire un modèle prédictif, pour une variable d'intérêt, à l'aide de variables explicatives (ou features). Pourtant, ces deux champs se sont développés en parallèle, créant ainsi deux cultures différentes. Le premier visait à construire des modèles probabilistes permettant de décrire des phénomèmes économiques alors que le second utilise des algorithmes qui apprennent sur la base de leurs erreurs, dans le but, le plus souvent de classer ou de prédire des quantités (des sons, des images, signaux, etc). Or ces dix dernières années, les modèles d'apprentissage se sont montrés plus efficaces que les techniques économétriques traditionnelles (avec comme prix à payer un moindre pouvoir explicatif). Plus spécifiquement, il apparaît qu'ils arrivent à gérer des données beaucoup plus volumineuses. Dans ce contexte, il devient alors nécessaire que les usagers des modèles linéaires comprennent ce que sont ces deux cultures, ce qui les oppose et surtout ce qui les rapproche, afin de s'approprier des outils développés par la communauté de l'apprentissage statistique, pour les intégrer dans des modèles économétriques et assurantiels.

L'utilisation de techniques quantitatives en économie remonte probablement au XVIième siècle, comme le montre Morgan (1990). Mais il faudra attendre le début du XXième siècle pour que le terme « économétrie » soit utilisé pour la première fois, donnant naissance à l'*Econometric Society* en 1933. Les techniques d'apprentissage automatique sont plus récentes. C'est à Arthur Samuel, considéré comme le père du premier programme d'auto-apprentissage, que l'on doit le terme « *machine learning* » qu'il définit comme « *a field of study that gives computer the ability without being explicitly programmed* ». Parmi les premières techniques, on peut penser à la théorie des assemblées de neurones proposée dans Hebb (1949) (qui donnera naissance au « *perceptron* » dans les années 1950, puis aux réseaux de neurones) dont Widrow and Hoff (1960) montreront quinze ans plus tard les liens avec les méthodes des moindres carrés, aux SVM (« *support vector machine* ») et plus récemment aux méthodes de *boosting*. Si les deux communautés ont grandi en parallèle, les données massives imposent de créer des passerelles entre les deux approches, en rapprochant les « deux cultures » évoquées par Breiman (2001a), opposant la statistique mathématique (que l'on peut rapprocher de l'économétrie traditionnelle, comme le note Aldrich (2010)) à la statistique computationnelle, et à l'apprentissage machine de manière générale.

Dans ce second chapitre, nous tentons de comprendre l'origine de l'utilisation quasi-systématique aujourd'hui de l'apprentissage statistique. Cela permet de formaliser correctement les problèmes de modélisation à partir de données. Selon l'objectif à atteindre, les approches peuvent alors se décliner en deux grandes familles d'algorithmes (ou de modèles) : supervisés et non supervisés. Selon la typologie de l'information traitée, les méthodes différent et sont d'autant plus facile à utiliser aujourd'hui qu'une multitude de librairies sont mises à disposition gratuitement aux utilisateurs : ce qu'on appelle communément l'*open source*. Nous présentons le fondement des algorithmes d'apprentissage statistique les plus communs (CART, réseaux de neurones, SVM, etc.) et les éclairons sous le regard des connaissances statistiques et économétriques.

La différence principale réside alors dans les hypothèses émises dans les deux disciplines. Toutes deux motivées par la compréhension d'un phénomène ou d'une tâche à partir de données, l'apprentissage statistique relâche les hypothèses probabilistes des modèles économétriques et étend les familles de fonctions à utiliser mais laisse en contrepartie plus de responsabilités aux utilisateurs sur le calibrage du modèle. Le risque de sur-apprentissage est donc plus grand et pose la question de la maîtrise du modèle tout au long de son utilisation. Les résultats séduisants peuvent cacher une erreur de décision dans la construction du modèle.

Une fois cette frontière bien identifiée, nous nous sommes focalisés sur le potentiel des algorithmes d'apprentissage profond très populaires depuis les années 2010. Plus précisément, nous avons étudié leurs applications aux traitements de données textuelles. Ce cas d'usage est en effet prometteur en assurance que ce soit pour le traitement automatique de pièces jointes, l'analyse de closes de contrat ou de vérification de dossiers. Ces travaux se déclinent alors au travers de différentes recherches et projets menés dans le cadre de la thèse CIFRE.

Les compagnies d'assurance accumulent en effet de nombreuses quantités de données qu'elles doivent parfois ingérer par l'intermédiaire d'une action manuelle. Dans le cas de la gestion de sinistre, cette intervention humaine peut alors rendre le processus de dédommagement long et coûteux. Par ailleurs, les nouvelles technologies et notamment les avancées en matière de parallélisation des calculs rendent aujourd'hui possible l'utilisation d'algorithmes complexes de *deep learning* présentant des performances attractives en matière d'analyse d'image et de texte. Automatiser le traitement de pièces associées à des sinistres constitue ainsi un service pouvant améliorer la réputation des assureurs.

Dans ce troisième chapitre, nous étudions les méthodes de réseaux de neurones profonds appliqués à la construction de ce nouveau service. Ce cas d'application tient son origine de la volonté pour Milliman d'être en mesure de proposer une solution concrète

pour le traitement automatique de documents numérisés (pdf, scannes, etc.). Nous étudions les méthodes d'analyse de reconnaissance optique permettant d'extraire à partir d'une image le texte qu'il contient. Il s'agit alors dans un premier temps d'extraire les zones d'une image contenant les lignes d'écriture puis les mots. Une fois cette zone isolée, nous nous sommes inspirés de la littérature et des techniques qui existent depuis les années 80 [Dupré 2004] afin de développer notre propre réseau de neurones. Après avoir assimilé différentes structures neuronales et leurs spécificités, nous avons établi une architecture par analogie avec la compréhension du mécanisme d'écriture : il existe des propriétés visuelles (connexion de pixels, courbures, etc.) mais également des dépendances temporelles dans l'écriture d'un mot (succession de lettres, ou de motifs).

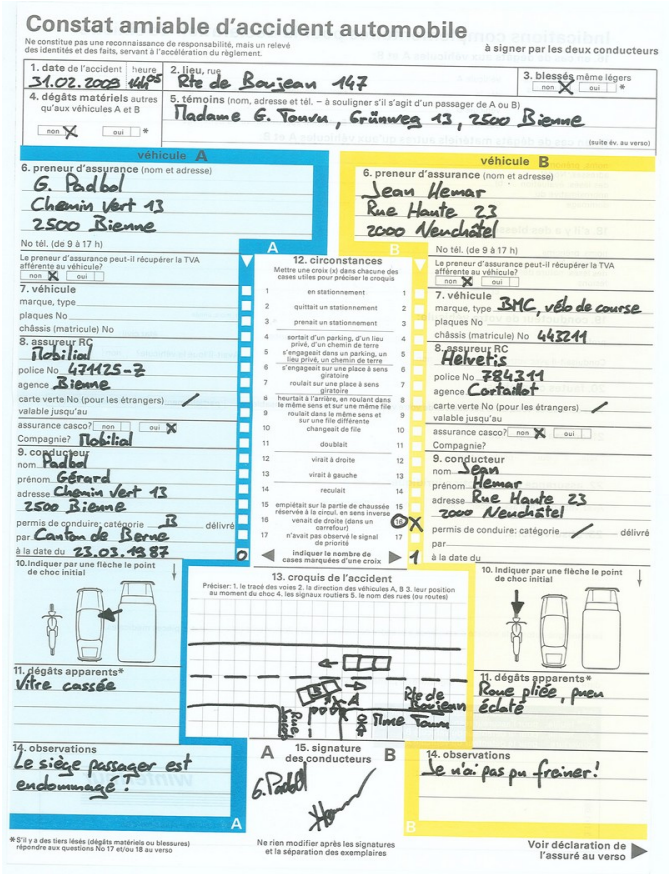


Figure 1.: Constat amiable rempli

En utilisant les techniques de parallélisation des calculs à partir des cartes graphiques [Owens et al. 2008], nous avons alors calibré le réseau de neurones résultant sur une base de données mise à disposition dans la littérature. Nous comparons enfin nos résultats avec l'état de l'art [Graves et al. 2009] avant d'appliquer un processus complet d'analyse

optique de caractères (OCR) à un constat d'assurance automobile amiable. Nos premiers résultats (dont un aperçu est présenté 2) mènent à vouloir poursuivre les travaux sur l'analyse d'écriture manuscrite.

Modèles	**CER (%)**	**WER (%)**
LSTM 30 [Pham et al. 2014]	14.72	42.03
LSTM 50 [Pham et al. 2014]	15.11	42.62
LSTM 100 [Pham et al. 2014]	15.79	44.37
LSTM 200 [Pham et al. 2014]	**14.68**	42.07
Our neural network	19.0	44.5
Our neural network with correction	17.11	**36.3**

Table 2.: Résultats obtenus sur la base de test

Les résultats obtenus sur la base de test Rimes sont assez satisfaisants, bien que plus faibles que ceux obtenus par Pham et al. (2014). En effet, même si le taux d'erreur moyen sur les mots (noté WER dans le tableau 2 et défini au chapitre 4) semble élevé, il est néanmoins à noter que l'on compte comme incorrecte une prédiction dès lors qu'au moins un des caractères n'a pas été prédit correctement. Le taux d'erreur dans la reconnaissance de caractères (noté CER dans le tableau 2 et défini au chapitre 4) indique qu'en moyenne 81% du mot prédit est par ailleurs correct. Si on regarde un échantillon de résultats comme illustré en figure 2, on remarque que les erreurs commises concernent des images où une lecture humaine est parfois difficile. Ainsi, mieux comprendre les fonctions d'OCR reste un sujet porteur que les modèles neuronaux semblent pouvoir résoudre que partiellement.

Figure 2.: Résultats de la reconnaissance de l'écriture manuscrite par les réseaux de neurones

Une fois le texte extrait, il a fallu s'intéresser au traitement automatique de cette information non-structurée. Ainsi, nous avons pu appréhender différentes techniques d'analyse sémantique. Le texte peut alors s'étudier de manière linguistique ou statistique. Si la première approche semble celle qui fut historiquement la plus répandue, les méthodes fréquentistes sont très vite devenues populaires.

La représentation simple d'une phrase ou d'un document est celle d'un histogramme. Chaque mot du dictionnaire représente une dimension et le document est représenté par un vecteur où chaque coordonnée informe de la fréquence d'apparition dans ce dernier. Cette représentation dite *sparse* car contenant beaucoup de coordonnées nulles, possède alors différentes variantes [Gupta et al. 2007] tenant compte de la fréquence d'apparition dans le document mais également dans le corpus permettant ainsi de ne pas sur-pondérer les pronoms ou mots non discriminants. Ces dernières années, les avancés de la recherche en *deep learning* ont permis d'établir un nouvel état de l'art en proposant une approche

de représentation des mots qui permet également de capturer le contexte des mots au sein d'une phrase ou d'un paragraphe : l'*embedding* Mikolov et al. (2013). La compréhension de certaines de ces techniques nous a permis d'accompagner différentes entreprises notamment dans l'utilisation des sources textuelles (commentaires, articles de presse, etc.)[8]. Ces travaux nous ont amené plus récemment à étudier les récentes publications des laboratoires de Google, Facebook et Microsoft. Ces derniers ont en effet franchi une nouvelle étape dans le traitement du texte et de la généralisation de la représentation textuelle à l'aide de l'algorithme BERT [Devlin et al. 2019] que nous explorons en fin de ce chapitre.

Même si les applications du *deep learning* permettent la création de nouvelles offres ou l'amélioration de services d'assurance préexistants, elles nécessitent cependant une grande quantité de données et une diversité suffisamment riche pour permettre aux algorithmes d'être performants. Plus généralement l'accumulation de données devient la source d'arbitrage entre précision des modèles prédictifs et protection des informations relatives aux individus. Si l'assurance est une industrie qui manipule des données personnelles depuis des dizaines d'années, la généralisation du *Big Data* à tous les secteurs amène le régulateur européen à imposer quelques limites dans l'utilisation des données et ce afin de protéger les citoyens.

Depuis mai 2018, le Règlement Général sur la Protection des Données (RGPD) s'applique à toutes les entreprises européennes. La législation renforce le contrôle des données à caractère personnel et leur stockage numérique. L'assurance devant transposer également le nouveau règlement, nous nous sommes intéressés à des méthodes de chiffrement alternatives compatibles avec les modèles de *machine learning* à l'image de *cryptonet* [Dowlin et al. 2016]. Ces méthodes n'étant pas encore matures et limitées par les capacités de calcul, nous nous focalisons alors dans ce chapitre sur l'impact que pourrait avoir l'anonymisation des données [Ly and Poinsignon 2019] sur les modèles de tarification non-vie. Plus précisémment nous proposons des méthodes d'anonymisation basées sur l'apprentissage non-supervisé qui permettent de préserver la précision moyenne de modèles de tarification automobile.

Cette étude commence par la compréhension des techniques d'anonymisation au regard des exigences de la réglementation. Ainsi, si les méthodes de chiffrement permettent de renforcer la sécurité en matière de stockage des données, elle n'empêche pas un manipulateur d'accéder à des données à caractère personnel. Par ailleurs l'anonymisation des données ne se restreint pas à la suppression des données sensibles mais se transcrit au travers d'un procédé rigoureux qui empêche toute assimilation d'un enregistrement à un individu. Après avoir défini les propriétés souhaitées d'un algorithme d'anonymisation, nous étudions l'utilisation des méthodes d'apprentissage statistique non-supervisées à

8 Ces projets ne sont malheureusement pas détaillés par soucis de confidentialité.

ces fins.

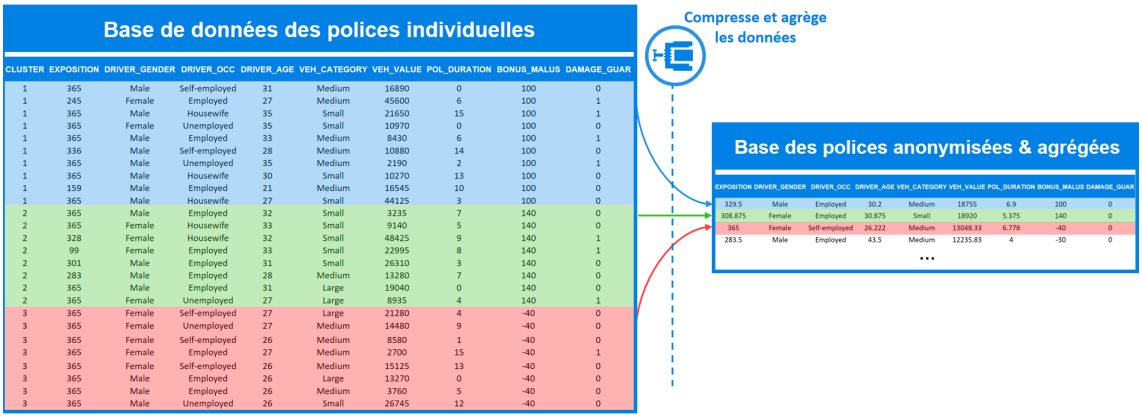


Figure 3.: Schéma de l'agrégation de polices d'assurances dans un contexte d'anonymisation

Plus précisément, nous avons considéré l'hypothèse où un assureur serait amené à anonymiser toutes les données de ses assurés passés. S'il souhaite continuer à exploiter l'information à des fins de tarifications, il doit alors, selon le RGPD, anonymiser ces dernières. Cependant, le processus de tarification nécessite de pouvoir préserver certaines propriétés des données afin de proposer un prix cohérent avec le profil de risque de l'assuré. Nous avons étudié les impacts que ces méthodes d'anonymisation par apprentissage non-supervisé pouvaient avoir sur un modèle de tarification automobile.

On constate que l'utilisation de K-means semblent être la moins impactante sur le modèle de tarification. Le prix obtenu sur les données ainsi anonymisées, n'est alors en moyenne que de $+4,6\%$ en écart relatif par rapport à la prime pure calibrée sur les données d'origine. La nature de la méthode de K-means s'appuyant sur une optimisation d'espérance permet sans doute d'expliquer ce résultat peu divergeant sur notre modèle de tarification qui repose également sur un calcul d'espérance.

L'analyse des écarts a été approfondie à la maille individuelle. Cela nous a notamment permis de mieux comprendre l'hypothétique impact de l'anonymisation sur les assurés [Ly and Poinsignon 2019]. Le graphique 4 permet par exemple de mieux comprendre les profils des assurés les plus impactés par une telle anonymisation. S'il faut croiser ces analyses avec les différentes expositions des profils dans la base, on peut néanmoins soulever qu'il existe une sur-représentation des catégories de retraités sur les lignes où le

modèle d'anonymisation a le plus d'écart.

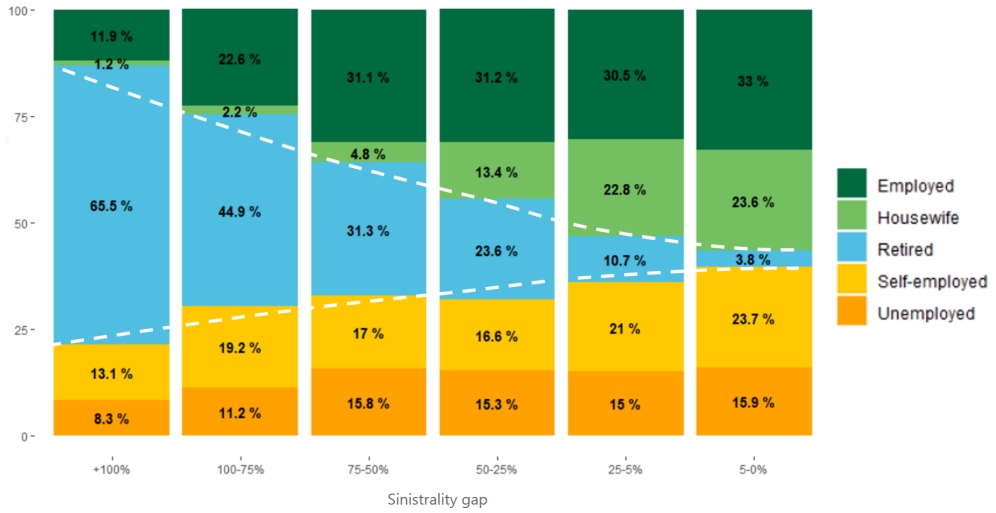


Figure 4.: Analyse des profils d'assurés selon les écarts avec le modèle ligne-à-ligne de référence

Ce constat nous a amené à lancer une nouvelle étude afin de mieux comprendre cet effet. Une anonymisation par agrégation avec un K-Means par exemple nous amène à sous-estimer les grands risques et sur-estimer les sinistres moyens plus nombreux. Cela s'explique alors assez naturellement par le fait que notre méthode effectue une mutualisation locale des risques lors de la construction de "l'individu" anonyme.

Conscients que notre étude est potentiellement sensible aux données que nous utilisons, nous mettons néanmoins en avant l'avantage d'utiliser certaines méthodes de classification afin d'impacter au minimum les modèles de tarification usuels (GLM). Nous proposons par ailleurs un premier ordre de grandeur d'impact sur la prime pure si cette anonymisation venait à être exigée.

Outre la question de l'anonymat des données, l'utilisation de plus en plus fréquente des modèles, certes performants, mais aux architectures complexes soulèvent également la question de l'interprétation des algorithmes. Qu'est-ce qu'un algorithme interprétable et qu'attend-on de ce dernier ? Pour répondre à cette question nous nous sommes alors plongés, dans ce cinquième chapitre, dans la littérature très récente sur le sujet. En effet, si des méthodes d'interprétation globale ont pu être introduites par J.H Friedman [Friedman 2001], ces dernières présentent de nombreuses limites. Elles ne tiennent en effet pas compte des interactions entre les variables ni des occurrences empiriques des modalités, menant à des interprétations contradictoires ou peu réalistes.

Ce n'est que depuis ces cinq dernières années [Apley 2014] que de nombreuses méthodes alternatives sont apparues. Des outils d'aide à l'interprétation globale mais également locale [Lundberg and Lee, Ribeiro et al. 2017, 2016] sont apparus afin de répondre aux questions de plus en plus nombreuses des régulateurs et institutions sur la transparence et l'interprétation des modèles. Ces dernières années, nous avons pu observer une croissance significative des publications sur ce sujet [Adadi and Berrada 2018]. Nous présentons alors un état des lieux des méthodes d'interprétabilité et exposons leur fonctionnement. On s'intéresse notamment à la définition aujourd'hui de ce terme et des limites des différents outils qui existent afin d'analyser des modèles assimilés à des boites noires comme le Gradient Boosting ou le Random Forest.

Ces outils d'interprétation comme SHAP ou LIME permettent d'éclairer le comportement de modèles à forte précision (comme on le constate empiriquement avec *Xgboost* ou les réseaux de neurones). Néanmoins, ils n'excluent pas une excellente compréhension des modèles. Par ailleurs, les comportements globaux sont encore limités. Ils reportent la notion d'interprétabilité à un autre niveau qui est celui des outils considérés pour comprendre le modèle complexe. Cette question ouvre alors la place à de nombreuses recherches qui suscitent un fort intérêt pour les entreprises.

Le *machine learning* est finalement une discipline qui ne cesse d'évoluer et ce de manière frénétique. Pouvant s'appliquer à toutes les industries, elle facilite l'échange des connaissances. Plus généralement la science des données semble indissociable de la communauté et de ses valeurs. Parmi elles, le partage des découvertes et la collaboration sont sans doute les plus notables. Elles expliquent certainement la rapidité avec laquelle ces dernières années la recherche en *machine learning* a pu avancer. Elles nous ont notamment permis de transposer certaines techniques au monde de l'assurance et à les observer parfois sous un regard actuariel. Il ne fut en effet pas rare d'utiliser en parallèle des recherches en *machine learning* systématiquement un logiciel de gestion de versions comme l'outil *git* afin de pouvoir accéder à l'implémentation associée aux différents papiers. Ces services permettent notamment de partager librement les codes ayant produit les résultats des différents travaux [9] et surtout laisse à la communauté la possibilité d'amélioration ou de correction de ces codes. En outre, le partage permet de favoriser les connexions entre les différents secteurs industriels mais surtout d'intensifier les échanges entre les écoles, laboratoires et les entreprises.

Cette thèse aura également été l'occasion de contribuer -à mon échelle- à la jonction de ces différents univers. Même si ce manuscrit n'en fait pas nécessairement état, au-delà des différents sujets sur lesquels j'ai pu travailler, j'ai contribué à l'organisation

9 Le site https://paperswithcode.com/ recense par exemple un grand nombre de papiers et des répertoires de code associés

de challenges internationaux : le Data Science Game[10] à destination des jeunes data scientists du monde entier. Ce challenge a été présenté lors d'un workshop de la conférence internationale NeurIPS [Ly et al. 2016]. L'organisation de cet événement consiste à organiser un hackathon de haut niveau rassemblant des équipes de quatre étudiants représentant tous leur université ou école. Depuis 2015, nous avons ainsi intéressés des milliers d'étudiants de tous les continents et rassemblé lors d'un événement final (cf www.datasciencegame.com) professionnels, académiques et entreprises afin de partager expériences et idées d'innovation à des problématiques réelles.

Ce soucis de vouloir relier les avancés de la recherche à l'industrie s'est également traduit tout au long de ma thèse par mes travaux chez Milliman. La maîtrise des différents algorithmes s'est aussi accompagnée d'un apprentissage des techniques de programmation (Python, R, Shell Linux) et de développement de logiciels (Continuous Integration (CI), Continuous Deployment (CD)). Pouvoir transformer un algorithme en un service à forte valeur ajoutée nécessite en effet d'appréhender la complexité des services informatiques. Nous avons pu enrichir notre connaissance sur le calcul distribué et nous familiariser notamment avec le framework Spark que nous avons déployé dans les bureaux de Milliman. Cette expérience est aujourd'hui un fort atout afin de comprendre l'intégralité des enjeux de l'application du *machine learning* dans le secteur de l'assurance. La maîtrise du calcul distribué me permet également aujourd'hui de transmettre cette connaissance aux étudiants de l'ENSAE Paris. Par ailleurs, l'utilisation des serveurs distants et l'usage du cloud nous ont permis de contribuer largement à l'étude menée par la SOA [Ly 2019] et également de travailler sur des sujets d'innovation comme l'utilisation des données télématiques en assurance automobile [Bellina and Ly 2018] bien que ces travaux ne soient pas détaillés ici.

Ces quatre années de thèse concluent ainsi mes débuts de recherche appliquée à l'industrie. Elles ouvrent par ailleurs de nombreuses perspectives. Les travaux menés plus spécifiquement sur l'anonymisation des données et leurs impacts ouvrent un nouveau champ d'exploration. De futurs travaux de recherche pourraient alors permettre de définir des relations entre les familles utilisées lors de l'approximation d'une fonction cible (comme un tarif) et la structure des données sous-jacente (impactée par la méthode d'anonymisation). Par ailleurs, la question de l'interprétabilité des modèles restent également ouverte, si des solutions existent, elles transposent actuellement la question sur l'utilisation d'autres modèles. L'utilisation de ces techniques appliquées à l'assurance par exemple remet la question du choix de la famille de modèles au cœur de la construction des nouveaux produits et de la maîtrise du risque. Enfin, si ces travaux nous ont permis de comprendre l'utilisation des réseaux de neurones, aucun fondement mathématique ne permet aujourd'hui de guider rigoureusement le choix de l'architecture d'un réseau. La plus grande difficulté reste sans doute l'assimilation des avancées toujours

10 datasciencegame.com

plus nombreuses et fréquentes qui devancent certainement la compréhension théorique de certains résultats empiriques.

2

UNE PREMIÈRE EXPLORATION DE L'APPRENTISSAGE STATISTIQUE EN ACTUARIAT

Dans les années 2010, l'apparition des objets connectés, des nouvelles données qu'ils engendrent ainsi que la multiplication des souscriptions d'assurance en ligne obligent les sciences actuarielles à repenser leurs modèles et à élargir leurs champs d'étude. En outre, dans ce contexte où la valeur des données augmente, des méthodes novatrices pour le monde de l'assurance : l'apprentissage statistique, présentent de nouvelles opportunités. Elles ouvrent ainsi un nouveau domaine de recherche actuariel aussi bien dans le cadre d'étude des comportements des assurés que dans le calcul de capital économique. En effet, depuis les différentes crises économiques et financières de cette dernière décennie, les institutions de réglementation imposent aux compagnies d'assurance de mieux se couvrir contre les risques futurs. Ainsi, la réglementation prudentielle Solvabilité II devant s'appliquer en janvier 2016 oblige les compagnies d'assurance à posséder suffisamment de fonds propres pour être capables d'empêcher toute ruine économique à un an avec une probabilité de 99,5%.

Néanmoins, la détermination de ce capital est souvent délicate. Alors que l'ancienne réglementation adoptait une approche comptable prudente, Solvabilité II propose une approche probabiliste plus complexe par l'estimation de *Best Estimate*. Même si l'EIOPA (*European Insurance and Occupationnal Pensions Authority*) propose une méthodologie formule standard, elle incite les grandes entités assurantielles à développer leur propres modèles dits internes afin de mieux s'adapter à leurs risques. Une méthode envisageable pour le développement d'un modèle interne est l'approche dite des Simulations dans les Simulations (SdS). Il s'agit par ce modèle, connaissant exactement les facteurs de risque d'une compagnie d'assurance, de simuler de nombreux scénarios économiques à l'aide de méthodes de Monte Carlo. On suppose ainsi couvrir tous les scénarios économiques pouvant survenir dans un an. Ces scénarios étant directement liés à la valeur des passifs d'une entreprise, on peut ajuster les fonds propres pour se couvrir contre les scénarios les plus risqués.

Mais cette méthode de référence est coûteuse en temps de calcul. La complexité de simuler la valeur des passifs tient au fait qu'en assurance il existe des interactions complexes entre passifs et actifs. Une simple formule de valorisation est souvent impossible

en assurance pour estimer la valeur d'un passif à un horizon futur. De plus, les méthodes simulatoires comme le SdS nécessitent des temps de calcul très importants (ces temps peuvent parfois atteindre la semaine de calcul). C'est pourquoi des méthodes alternatives sont apparues comme les *Replicating Portfolios* [Devineau and Chauvigny 2010] ou plus généralement les méthodes d'approximation par *Least Squares Monte Carlo* [Koursaris 2011]. La dernière méthode consiste par exemple à construire une forme paramétrique à partir de différents facteurs de risque afin d'approcher la valeur économique de la compagnie d'assurance. Son avantage est de permettre, à l'issue de chaque scénario monde réel de première période, une évaluation rapide et simple, de la valeur des passifs. Seulement, une erreur d'approximation peut avoir de lourdes conséquences sur le bilan d'une compagnie d'assurance. Il est donc important d'adopter un modèle robuste et peu biaisé. Parce que les méthodes polynomiales aujourd'hui utilisées en *Least Squares Monte Carlo* représentent toujours un enjeux dans la détermination du capital économique nous avons donc souhaité étudier une approche nouvelle par l'apprentissage statistique.

Avec l'intérêt croissant des assureurs pour les méthodes d'apprentissage statistique[1], nous souhaitons les étudier dans un cadre plus élargi. Ce premier chapitre propose ainsi au travers de l'étude des réseaux de neurones de mettre en évidence les enjeux opérationnels et théoriques du *Machine Learning* mais aussi certaines limites. Plus largement, ces travaux permettent de mieux comprendre dans quelle mesure l'apprentissage statistique permet d'élargir le panel d'outils dont dispose aujourd'hui l'assureur pour répondre aux exigences du régulateur mais également de maîtriser le risque en étudiant les comportements des assurés. Nous comparerons ainsi les performances des réseaux de neurones à des modèles utilisés traditionnellement par les assureurs.

Si pour la première application, nous avons choisi de nous intéresser à l'assurance vie, c'est que de nombreux travaux se focalisent sur l'application du *Machine Learning* à des problématiques d'assurances non-vie [Lopez et al. 2015] [Milhaud 2011]. En assurance vie, les méthodes d'apprentissage statistique se concentrent aujourd'hui sur des problématiques de classification : détermination des scénarios à risque [Garnier and Martial 2013] [Niedzwiedz 2014] ou encore risque d'arbitrage de contrats d'assurance vie [Guillot 2014] [Milhaud 2011]. En ce qui concerne les réseaux de neurones, bien qu'ils semblent très prisés par les GAFAM[2], ils apparaissent très peu à ce jour dans la littérature actuarielle. Dans les mémoires recensés[3] qui en font référence, leurs études aboutissent pourtant à un bon pouvoir prédictif lorsque le modèle est entraîné sur des historiques de données réelles [Richard 2013] [Aouizerate 2010]. Ces résultats nous ont donc amené à nous demander dans quelle mesure cette méthode pouvait être performante dans la

1 Connues aussi sous le terme anglais *Machine Learning*
2 Google, Amazon, Facebook, Apple, Microsoft
3 http://www.ressources-actuarielles.net/

détermination d'un capital économique à un an dans un cadre Solvabilité II.

Cette étude innove pour deux raisons : il est assez peu courant en assurance vie[4] d'utiliser l'apprentissage statistique et surtout, les réseaux de neurones sont étudiés non pas uniquement sur des observations réelles mais sur des données financières simulées. D'autre part, l'utilisation des modèles neuronaux dans un cadre de régression (et non de classification) est peu courante. Les travaux de ce chapitre permettent alors d'en savoir un peu plus quant à l'application des neurones formels à une problématique nouvelle où l'utilisation de régression linéaire est répandue. Tout en ayant un regard critique actuariel sur nos résultats, nous adoptons dans ce qui suit une approche de *datascientist*. Ainsi, il apparaît important de mettre également en avant le côté opérationnel (informatique) indissociable de l'application des méthodes d'apprentissage statistique qui représente une discipline nouvelle pour l'actuaire.

Nous commencerons donc par introduire le contexte actuariel de notre problématique qu'est l'utilisation des réseaux de neurones à des fins de détermination de capital économique. Plus précisément, nous verrons dans quelle mesure cette méthode présente une alternative à l'approche polynomiale utilisée en *Least Squares Monte Carlo* [Koursaris 2011]. Ensuite, après avoir présenté le modèle neuronal nous étudierons ses performances et approfondirons l'analyse du modèle.

4 Cette partie a été rédigé en début de thèse. Le côté "innovant" 3 ans plus tard peut sembler inadapté.

2.1 ETAT DE L'ART DE LA DATA SCIENCE EN ASSURANCE

2.1.1 *L'actuaire datascientist : une nouvelle approche, des nouveaux enjeux*

Dans une société de plus en plus numérisée et avec l'apparition des objets connectés, les données ont pris une importance cruciale. Loin d'échapper au phénomène, l'assureur (et donc l'actuaire) doit s'adapter à ce nouvel environnement où les données sont de plus en plus abondantes et facilement accessibles. Dans cette partie, nous tentons alors de mettre à jour les différentes disciplines qui viennent élargir le panel de compétences nécessaires à l'actuaire de demain. Ces points permettront de mieux comprendre les nouveaux enjeux qu'amène l'utilisation de l'apprentissage statistique en actuariat.

Comment définir un datascientist?

«La science des données est une nouvelle discipline qui s'appuie sur des outils mathématiques, de statistiques, d'informatique. Cette science est principalement une "science des données numériques" et de visualisation des données »[5]. Selon cette définition, la *data science* est une discipline hybride nécessitant des compétences aussi bien théoriques qu'informatiques. Il est important de souligner que ce terme ne signifie pas nécessairement l'implication d'utilisation de *Big Data*.

En effet, le terme de *Big Data* fait référence à la volumétrie des données. Ainsi, il désigne plus largement les infrastructures informatiques et les systèmes logiciels mis en œuvre dans la manipulation des données massives. C'est parce que la volumétrie des données et les nouvelles technologies qui les gèrent ont amené les scientifiques à repenser leurs algorithmes que ce terme est aujourd'hui abusivement associé à l'apprentissage statistique. On retiendra tout de même qu'un enjeu pour l'actuaire *datascientist* est d'être en mesure de manipuler la donnée qu'elle soit massive ou non. Il faut donc se familiariser avec différents langages et algorithmes parfois nouveaux en science actuarielle.

Les sciences actuarielles sont aujourd'hui considérées comme une branche des mathématiques appliquées. Ainsi, les statistiques, les mathématiques financières ou encore les théories microéconomiques sont couramment utilisées. Les *datasciences* introduisent alors une approche nouvelle à ces différents outils maîtrisés par l'actuaire. Loin de les révolutionner elles abordent parfois les mêmes problématiques mais sous des contraintes différentes. Ainsi l'apprentissage statistique introduit non seulement des algorithmes nouveaux (Réseaux de neurones, SVM, Arbres de Classification et Régression, kNN etc.) pour la science actuarielle mais également des outils nouveaux permettant de s'adapter et manipuler les nouvelles données (R, Python, Spark ou Hadoop MapReduce pour les données massives etc.).

5 Source Wikipédia

Quelles sont les outils les plus utilisés ?

Il existe de nombreux outils permettant d'effectuer de l'apprentissage statistique. Dans la communauté des *datascientists* les logiciels *Open Source* sont particulièrement prisés. Ils profitent en effet d'une actualisation permanente des algorithmes et permettent ainsi de bénéficier des dernières recherches à moindre coût. Parmi ces logiciels, deux se distinguent[6] par leurs nombres d'utilisateurs : Python et R.

R est langage «Script» , largement utilisé par la communauté des statisticiens. R permet de produire des visualisations agréables. Logiciel d'origine statistique, il incorpore de nombreux algorithmes de *Machine Learning*. Néanmoins, ce langage n'est pas nécessairement optimisé et donc présente des temps de calculs plus long que d'autres langages. R bénéficie d'une grande communauté de contributeurs de packages qui permet au logiciel de s'adapter à la structure des nouvelles données et des nouvelles infrastructures informatiques.

Python est un langage «Orienté Objet», largement utilisé par la communauté des informaticiens et des mathématiciens. Le package *scikit-learn* de *machine learning* propose de nombreux algorithmes d'apprentissage statistique. Ce langage modulaire est adapté au développement d'outils. Il facilite également les suivis et les échanges grâce à l'interface Ipython Notebook, qui permet de partager et documenter facilement des travaux sans aucun logiciel (publication de pages web). Cependant, le logiciel n'égale pas encore la richesse des packages de R en statistique même si l'écart s'est considérablement réduit.

Comme nous le détaillerons par la suite, nos premiers travaux s'intéressent à l'utilisation de réseaux de neurones. Dans cette optique, il est apparu au cours de nos recherches (en 2015) que ce type de modèle était plus facilement accessible sous le langage python[7]. Il est toutefois difficile de prétendre qu'un logiciel est "meilleur" qu'un autre. Plutôt que de chercher à les classer, nous avons choisi le langage le plus adapté à notre besoin. D'autre part, dans un souci de compréhension du modèle et de son développement, nous avons utilisé un modèle codé par nos soins. Le langage orienté objet étant approprié aux réseaux de neurones nous avons utilisé le langage Python. Nous présenterons ces

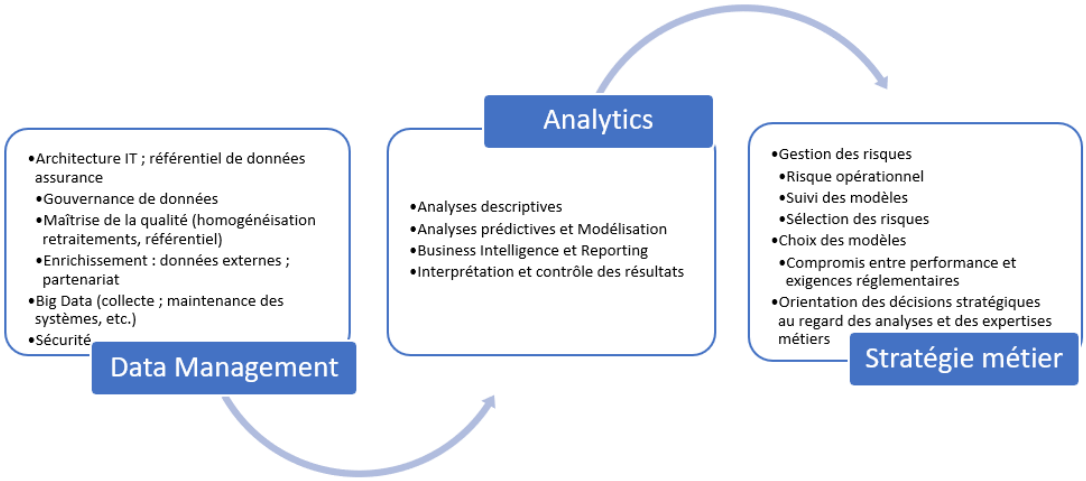
6 Il existe de nombreux autres logiciels non mentionnés ici.

7 Sous R, les packages *nnet*, *RSNNS*, *neuralnet* existent mais semblaient peu commodes à notre étude, sous Python les packages *Theano* et *PyBrain* sont très complets et permettent la parallélisation des calculs. En 2019, la quasi totalité des librairies de *deeplearning* sont interfacées avec Python

travaux plus loin dans ce chapitre.

Un actuaire, de par sa formation possède des compétences en statistique et est familier avec de nombreux outils mathématiques. Là où réside l'enjeu est alors dans la maîtrise des langages informatiques, dans l'approche spécifique d'un problème en apprentissage statistique et dans les algorithmes permettant d'exploiter différemment les nouvelles données mises à sa disposition dans un contexte réglementaire défini aussi bien par la CNIL[8] que par les autorités de régulation des assurances. En définitive, l'abord d'un problème d'apprentissage statistique dans un environnement assurantiel peut se décomposer en trois axes qu'illustre le schéma de la Figure 5.

Figure 5.: Chaine de valeur dans l'exploitation des données



Une première partie -des plus importantes et des plus longues à mettre en place- consiste à collecter et nettoyer les données brutes. Cette phase nécessitant des compétences dans la manipulation des infrastructures *Big Data* et des outils de traitement de données amène ensuite à une analyse permettant de répondre à une problématique. Dans cette deuxième phase, la modélisation et la méthode de *Machine Learning* à mettre en œuvre sont abordées. Enfin, dans la dernière partie, les résultats sont intégrés dans la détermination de la stratégie métier. Selon les contraintes de l'entreprise les deux premières phases permettent alors en complément des analyses actuarielles déjà existantes de gérer par exemple un risque ou de mettre en place une action commerciale afin d'anticiper un comportement de rachat ou d'arbitrage.

8 Commission Nationale de l'Informatique et des Libertés

2.1.2 *Une stratégie métier encadrée par Solvabilité 2*

Avant de présenter une étude de risque comportementale qui explore la totalité de la chaîne de valeur de l'exploitation des données, notre première analyse se place dans le troisième cadre de la Figure 5. Notre premier cas d'étude évaluera la capacité d'un modèle d'apprentissage statistique à déterminer un capital économique dans le cadre particulier de Solvabilité II. En s'appuyant sur des données simulées, notre premier cas pratique s'affranchit alors en grande partie de la préparation des données, le type de modèle étant également choisit en amont (seul le calibrage du modèle doit être mené), il reste alors à évaluer la conformité des résultats. Nous verrons par la suite une étude qui illustrera l'intégralité d'une démarche à laquelle un actuaire utilisant des outils de *data science* peut être confronté.

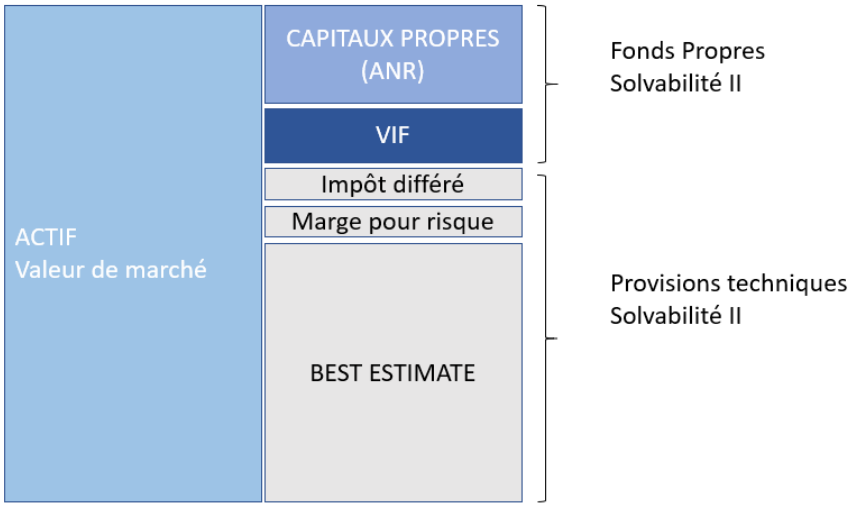
Solvabilité II est un projet de réforme européenne. Dans la lignée de Bâle II pour le secteur bancaire, cette réglementation prudentielle a pour objectif d'assurer une bonne évaluation des risques auxquels les compagnies d'assurance sont soumises. Ces dernières doivent en particulier déterminer leur capital économique afin de pouvoir faire face à une ruine économique dans un an avec une probabilité de 99,5%. Mathématiquement, ce capital est déterminé par la formule suivante :

$$C = FP(0) - P_{ZC}(0,1) \times q_{0.5\%}(FP(1)) \tag{1}$$

où C désigne la valeur du capital économique, $FP(t)$ la valeur des fonds propres à la date t (en années), $q_{0,5\%}$ désigne la fonction quantile à $99,5\%$ et où $P_{ZC}(0,1)$ désigne le prix d'un zéro coupon de maturité 1 an observé à la date $t = 0$, il constitue le facteur d'actualisation.

Afin d'illustrer la définition des fonds propres dans un bilan comptable, le schéma ci-dessous expose une description simplifiée du bilan économique selon la réglementation Solvabilité II.

Figure 6.: Vision comptable simplifiée du bilan économique SII - source Milliman, ANR: Actifs Nets Réévalués, VIF: Value In Force (mesure de rentabilité)



Par souci de simplification, nous assimilerons la valeur économique des passifs au *Best Estimate* ; "meilleure estimation" en français, il correspond à la valeur actuelle nette probable des flux futurs (prestations, frais, commissions, primes) selon la définition fournie par la réglementation Solvabilité II [CEA 2008]. Nous ne prendrons donc pas en compte, dans la suite de ce rapport, la marge pour risque[9] qui devrait être ajoutée au *Best Estimate*.

Remarque : par équilibre comptable on remarque que, connaissant à un instant t donné, la valeur des actifs $A(t)$, afin de calculer le capital économique, il est équivalent d'évaluer directement $FP(t)$ (avec les notations de l'équation (1)) et d'évaluer le *Best Estimate* $BE(t)$. Pour l'instant t, par équilibre comptable, on dispose en effet de la relation :

$$A(t) = BE(t) + FP(t). \tag{2}$$

Deux types d'approches sont alors possibles dans la détermination de ce capital (soit des fonds propres) : la formule standard proposée par l'EIOPA (*European Insurance and Occupationnal Pensions Authority*) ou l'utilisation d'un modèle interne parfois plus adapté aux risques inhérents aux entreprises. Dans cette partie nous introduisons la dernière méthode avant de présenter une technique d'élaboration de modèle interne : la méthode des *Least Squares Monte Carlo*.

9 Provision complémentaire s'ajoutant au *Best Estimate* dont le calcul est fixé par la réglementation

L'utilisation d'un modèle interne

L'évaluation des fonds propres à un an permet de faire face à une ruine économique probable à 0.5%. Cependant, mobiliser trop de capital peut être coûteux pour la compagnie d'assurance. En effet, les différents investisseurs exigent souvent une rémunération de leurs placements dans la compagnie. Ainsi, bien que prudente, une sur-évaluation des fonds propres n'est pas nécessairement bénéfique pour l'entité. C'est la raison pour laquelle il est préférable pour certaines structures de développer leur propre méthodologie d'évaluation du capital de solvabilité requis.

Cette méthodologie s'appuie non pas sur la formule standard proposée par l'autorité de régulation mais est développée par les équipes de recherche de la compagnie. Le calcul de capital économique se fait à partir d'un modèle propre à l'entité. Il tient compte des spécificités financières et structurelles de l'entreprise. En prenant en compte les risques qui lui sont propres (que l'on appellera facteurs de risque[10]), l'assureur est alors en mesure de capitaliser à la hauteur de son besoin.

Seulement, la connaissance du capital exacte requis pour faire face à une ruine économique à un an est impossible. Des méthodes simulatoires permettent néanmoins d'approcher cette valeur. Comme tout estimateur statistique, l'échantillonage introduit nécessairement un biais qu'il est préférable de minimiser. Mais cela représente un coup opérationnel qui rend délicate l'application des méthodes théoriques idéales comme les Simulations dans les Simulations.

La sélection des facteurs de risques

En assurance vie, chaque compagnie est soumise à un ensemble de risques : financiers, opérationnels, techniques, etc. Comme dans toute modélisation interne, il est nécessaire de les identifier dans un premier temps. Dans notre présentation de la méthode des Simulations dans les Simulations, nous considérons uniquement des facteurs de risque financiers[11]. Par exemple, ces risques financiers peuvent être des risques de taux, des risques liés à un investissement en action ou encore des risques liés à l'achat de produits dérivés comme des swaptions. Chacun des risques financiers introduit donc un aléa dans la valeur future du portefeuille d'investissement de l'assureur qui on le rappelle, permet d'équilibrer son bilan comptable c'est-à-dire de lui permettre de faire face à ses engagements envers ses assurés. Cet aléa sera par la suite appelé facteur de risque que ce soit dans le cas du *Least Square Monte Carlo* présenté plus loin ou des Simulations dans les Simulations.

10 Ces facteurs de risque peuvent être financiers, opérationnels démographique etc.

11 La méthode s'étend en considérant des facteurs de risque techniques qui conditionnent alors les simulations secondaires et la détermination des *Best Estimate*

2.1.3 *La méthode des Simulations dans les Simulations*

La méthode des Simulations dans les Simulations (SdS) est une technique utilisée essentiellement dans un cadre d'assurance vie permettant d'estimer une distribution des fonds propres économiques à un an. Ce modèle permet ainsi à l'assureur d'augmenter au besoin la valeur de ses fonds propres pour être en mesure de faire face à une ruine économique avec une probabilité de 99,5 %. Dans la suite de ce mémoire, nous considérons uniquement des facteurs de risque financiers. La méthode des Simulations dans les Simulations se décomposent alors en trois étapes majeures :

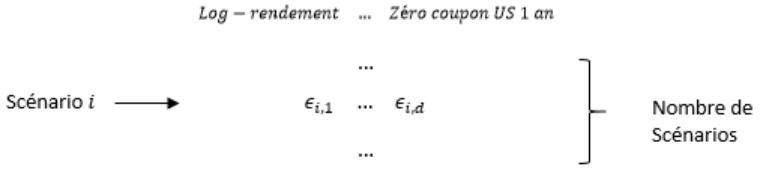
1. La simulation en première période de différents scénarios économiques;
2. Pour chaque scénario de première période, la simulation de nouveaux scénarios conditionnés par les conditions économiques établies par la simulation primaire;
3. Pour chaque trajectoire de simulation primaire, l'injection de la table de simulations secondaires dans un modèle de gestion actifs-passifs afin de fournir la valeur des fonds propres associée au scénario primaire.

2.1.3.1 *La Simulation en première période*

On introduit la formalisation suivante :

Soit $d \in \mathbb{N}^*$, soit $\epsilon = (\epsilon_1, ..., \epsilon_d)$ le vecteur aléatoire (i.e. vecteur des facteurs de risques) de loi de probabilité $\mathbb{P}$. On notera $\epsilon_i \in \mathbb{R}^d$ le vecteur des $d \in \mathbb{N}^*$ facteurs de risques associés à la compagnie pour un scénario $i \in [\![1, P]\!]$ si $P \in \mathbb{N}^*$ désigne le nombre de scénarios primaires. Le schéma de la Figure 7 illustre les notations que nous utiliserons par la suite.

Figure 7.: Exemple d'un vecteur de facteur de risque $\epsilon = (\epsilon_1, ..., \epsilon_d) \in \mathbb{R}^d$



Une fois les facteurs de risques identifiés, la méthode consiste à simuler différents scénarios économiques à 1 an. Dans notre cas où l'on considère uniquement des facteurs de risque financiers, ces simulations s'effectuent en général sous la probabilité historique (appelée également monde réel) que nous avons noté $\mathbb{P}$. Les simulations doivent d'autre

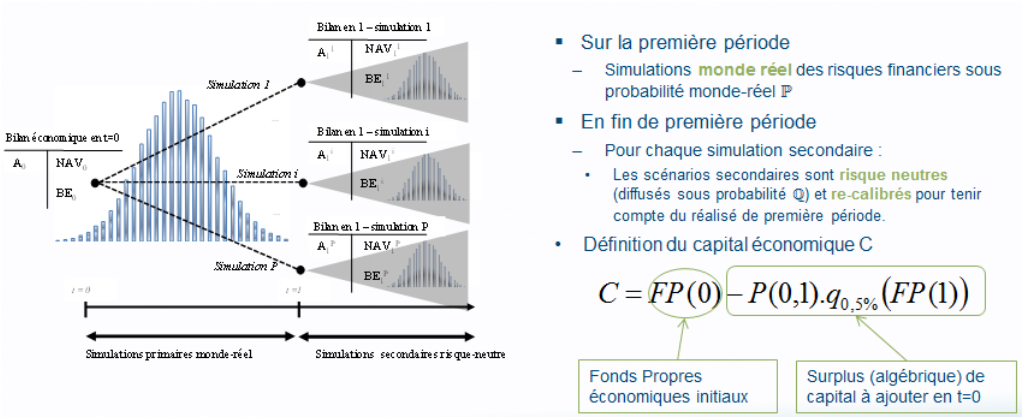
part couvrir un large panel de scénarios probables et réalistes. On dispose ainsi en première période d'un large nombre de situations économiques simulées sous une probabilité historique. Nous noterons $P \in \mathbb{N}^*$ le nombre de simulations primaires. C'est-à-dire le nombre de scénarios économiques projetés à un an.

2.1.3.2 *La Simulation de deuxième période*

Une fois les scénarios économiques de première période simulés, les valeurs à 1 an des fonds propres de la compagnie restent à évaluer. Pour se faire il est courant d'effectuer une deuxième simulation sous une probabilité risque neutre notée $\mathbb{Q}$ des facteurs économiques afin de pouvoir déterminer les flux probables générés par les passifs. Ce choix de probabilité risque neutre est motivé par son utilisation courante en finance (sous l'hypothèse de non-arbitrage) pour la valorisation des actifs.

Si les flux engendrés par les actifs se déterminent facilement à partir du scénario économique établi par la trajectoire de première période, les flux financiers engendrés par les passifs nécessitent la génération d'une table de scénario secondaire.

Figure 8.: Construction de la totalité de la distribution des Fonds Propres économiques de fin de première période, puis déduction du capital économique - source Milliman, NAV: Net Asset Value



2.1.3.3 *Valorisation et détermination de la distribution de Best Estimate*

Enfin, les passifs d'une compagnie d'assurance (par exemple les contrats d'assurance épargne vie) et les actifs financiers sont étroitement liés. La détermination des fonds propres à un an nécessite alors l'utilisation d'un modèle de gestion d'actifs et passifs [12]. En outre, ce modèle tient compte par exemple de l'évolution du risque de rachat,

12 Souvent désigné par l'acronyme anglais ALM (Assets and Liabilities Model)

l'évolution du nouveau taux de rendement de l'épargne des assurés en fonction des nouvelles conditions économiques en date $t = 1$.

Lorsque le nombre de scénarios risque neutre est alors élevé ce modèle devient long à exécuter et rend alors peu opérationnel la détermination des flux de fonds propre pour chacune des trajectoires primaires simulées. En pratique, d'après l'article de L.Devineau et S.Loisel, cela contraint les compagnies d'assurance à n'effectuer qu'une centaine de simulations primaires pour environ un millier de simulations secondaires [Devineau and Loisel 2009]. L'objectif étant de déterminer au final les fonds propres nécessaires pour se couvrir contre un risque de ruine à un an avec une probabilité de 99,5%, ne posséder qu'une centaine de scénarios n'est pas idéal dans la mesure où la distribution est alors peu riche [EIOPA 2014].

2.1.4 *Le curve-fitting*

Malgré les contraintes opérationnelles qu'imposent la réalisation d'une génération d'une distribution de fonds propres à un an par la méthode des Simulations dans les Simulations, une première approche consiste à supposer qu'il existe une fonction mesurable g de sorte que

$$BE = g(\epsilon).$$

La méthode de *curve-fitting* consiste donc à partir de quelque valeurs de fonds propres à un an générées par la méthode des Simulations dans les Simulations (le maximum qu'autorisent les infrastructures dans le temps disponibles - en pratique quelques centaines) de calibrer une fonction g dépendant des facteurs de risque ϵ. Actuellement, le calibrage se fait au travers d'une régression linéaire. Les réseaux de neurones présentent ainsi une alternative que nous étudions.

Une des difficultés est souvent de conserver un rapport raisonnable entre la dimension du vecteur d'entrée et le nombre de d'observations de BE que nous disposons. Il est en effet possible de considérer des puissances des facteurs de risque initiaux ainsi que des termes croisés afin de pouvoir capter dans la constitution d'un polynôme différentes interactions entre les aléas.

Enfin, même si la méthode des Simulations dans les Simulations est une référence dans la détermination de *Best Estimate*, elle ne fournit qu'une estimation plus ou moins précise en fonction du nombre de simulations secondaires. Si l'on souhaite générer une quantité plus importante de *Best Estimate*, cela se fait alors au détriment du nombre de simulations secondaires rendant les valeurs de BE plus bruitées. Nous tenterons dans la suite de ce mémoire de formaliser l'influence de l'échantillonnage (c'est-à-dire le nombre de simulations secondaires) sur l'erreur d'estimation.

2.1.5 *Une méthode alternative : le Least Squares Monte Carlo*

Pour faire face à la complexité[13] de la méthode SdS, certaines compagnies d'assurance utilisent une méthode simulatoire alternative que l'on peut assimiler à l'adaptation des régressions économétriques aux données simulées par *Monte Carlo* décrite par exemple dans le livre de Bruno Dupire (1998) "Monte Carlo:methodologies and applications for pricing and risk management".

Chaque scénario monde réel $p \in \mathbb{N}^*$ peut être représenté par un vecteur de facteurs de risque monde réel que nous noterons par la suite ϵ_p. Ainsi, si $d \in \mathbb{N}^*$ représente le nombre de facteurs de risque (indépendant du scénario considéré) simulés sous la probabilité historique $\mathbb{P}$, alors ϵ_p est un vecteur de $\mathbb{R}^d$ où chaque composante représente la valeur prise par un facteur de risque[14] pour un scénario donné. On appellera l'entier d dimension du *Least Squares Monte Carlo*.

Par définition, la valeur économique des passifs BE_1^p après un scénario monde réel p correspond à la valeur espérée (sous la probabilité risque neutre $\mathbb{Q}$) actualisée des passifs en $t = 1$. En notant VAN cette variable aléatoire (Valeur Actuelle Nette), cette définition se transcrit mathématiquement par l'équation :

$$\mathbb{E}^{\mathbb{Q}}(VAN \mid \mathcal{F}_1^p) = BE_1^p,$$

où $\mathcal{F}_1^p$ est la tribu associée à l'information disponible des marchés financiers à la date $t = 1$.

Alors que le calcul d'une valeur de Best Estimate BE après un scénario monde réel nécessite un jeu complet de scénarios risque neutre, le calcul d'une valeur de VAN ne nécessite lui qu'un seul scénario risque neutre par définition.

Si on fait l'hypothèse faible et presque toujours vérifiée que l'information du marché à $t = 1$ est contenue exactement dans le vecteur ϵ , alors l'équation précédente se réécrit :

$$\mathbb{E}^{\mathbb{Q}}(VAN \mid \epsilon_p) = BE_1^p.$$

Ainsi, BE_1 apparaît comme une fonction mesurable dépendant de ϵ et peut donc se mettre sous la forme :

$$\mathbb{E}^{\mathbb{Q}}(VAN \mid \epsilon) = BE_1 = f(\epsilon) \tag{3}$$

13 Au sens computationnel

14 A titre d'exemple, le log-rendement d'un indice de référence entre $t = 0$ et $t = 1$ ou les variations du niveau d'une courbe d'intérêts entre $t = 0$ et $t = 1$ sont des facteurs de risques standards.

où $f : \mathbb{R}^d \to \mathbb{R}$ est une fonction mesurable et unique (presque sûrement).

Contrairement au *curve-fitting* dans cette méthode, f est calibrée à partir de l'observation de VAN et non directement du *Best Estimate* (BE). En outre, on fait l'hypothèse qu'il existe une variable aléatoire réelle μ tel que :

$$VAN = f(\epsilon) + \mu$$

avec $\mathbb{E}^{\mathbb{Q}}(\mu \mid \epsilon) = 0$. La méthode de Least Square Monte Carlo consiste donc à estimer la valeur d'un BE en générant différentes valeurs de VAN chacune associée à un vecteur de scénario primaire ϵ. On remarque ainsi qu'il n'est plus nécessaire de générer un jeu complet de scénario secondaire pour chaque scénario primaire ϵ. Comme le fait remarquer Bruno Dupire (1998), cette approche se fait alors au détriment de la variance de l'estimateur du BE obtenu.

Un premier axe de recherche réside donc dans la manière de déterminer f. Si f est continue, une approche possible est d'utiliser le théorème de Stone-Weierstrass et d'approcher f par un polynôme. Une autre, plus innovante en assurance, consisterait à utiliser l'apprentissage statistique pour déterminer f.

2.1.6 *Un nouveau regard : l'apprentissage statistique*

L'apprentissage statistique a pour objet des procédures automatiques permettant d'approcher le fonctionnement inductif du cerveau humain. Bien que cette science soit affiliée à l'informatique, elle est étroitement liée [Dalayan 2014] à la statistique non-paramétrique. Néanmoins, là où la statistique se concentre sur les modèles, l'apprentissage statistique lui, se focalise plutôt sur les algorithmes. Un des avantages de cette approche est qu'elle ne nécessite pas d'*a priori* sur la distribution des données.

Dans un cadre général, on introduit une base de données de $n \in \mathbb{N}^*$ observations comme l'ensemble des n couples (X_i, Y_i) distribués selon une probabilité $\mathbb{P}$. Où pour tout $i \in [\![1, N]\!]$, X_i appartient à $\mathcal{X}$ (espace vectoriel de dimension finie) appelé espace d'entrée et Y_i appartient à $\mathcal{Y}$ appelé espace de sorties ou étiquettes. On conservera ces notations pour la suite.

2.1.6.1 *Deux grandes familles d'algorithmes*

Selon l'espace $\mathcal{Y}$ des variables à expliquer, et également selon le type de problématique, on distingue deux grandes familles d'apprentissage statistique :

1. L'apprentissage non-supervisé : Les données Y_i ne sont pas disponibles. Cette famille d'algorithmes cherche à alors déterminer la loi de probabilité $\mathbb{P}$ ayant généré les observations X_i. En actuariat ces méthodes sont utilisées afin de regrouper les différentes observations X_i en groupes homogènes (définies selon un critère - en général une distance euclidienne ou de khi-deux). L'étude des différentes classes permet alors de faire ressortir des caractéristiques identifiant une tendance majeure au sein du groupe. On peut ainsi affecter un label artificiel Y_i (où Y_i est par exemple le numéro de la classe déterminée par l'algorithme) à chaque observation X_i. Par exemple, les méthodes de classifications à ascendance hiérarchique appartiennent à ce type d'algorithme. La difficulté de ce type d'algorithme provient du fait que l'on ne possède *a priori* aucune information sur la classe "réelle" de l'individu / observation.

2. L'apprentissage supervisé : à partir d'une base de données de $n \in \mathbb{N}^*$ observations (X_i, Y_i) cette famille d'algorithmes a pour objectif d'apprendre à prédire l'étiquette $y \in \mathcal{Y}$ en fonction d'une entrée $x \in \mathcal{X}$. Au travers de cette définition, on peut remarquer que les méthodes économétriques usuelles peuvent s'étudier comme des méthodes d'apprentissage supervisé. Seulement il est important de souligner qu'en apprentissage statistique, aucune hypothèse sur la distribution des données n'est nécessaire en général.

Dans la suite de ce chapitre, nous nous intéressons plus particulièrement à la famille des méthodes d'apprentissage supervisé à laquelle les réseaux de neurones appartiennent.

2.1.6.2 *Approche générale de l'apprentissage supervisé*

Comme introduit dans le paragraphe précédent, le but de l'apprentissage statistique supervisé est de trouver la meilleure fonction de prédiction $f^* \in \mathcal{F}(\mathcal{X}, \mathcal{Y})$. De sorte que :

$$f^* \in arg \min_{f \in \mathcal{F}(\mathcal{X}, \mathcal{Y})} \mathbb{E}_{\mathbb{P}}[l(Y, f(X))]$$

où $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ est appelée fonction de coût. On appelle également le terme $\mathbb{E}_{\mathbb{P}}[l(Y, f(X))]$ risque de la fonction de prédiction f et on le note $R_{\mathbb{P}}(f)$.

Cependant, le nombre d'observations est limité. Ainsi, en pratique, nous estimons un risque empirique sur $n \in \mathbb{N}^*$ observations :

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} l(Y_i, f(X_i)).$$

En partant de l'hypothèse que $\mathbb{E}_{\mathbb{P}}[l(Y, f(X))^2] < +\infty$, la loi forte des grands nombres et le théorème centrale-limite permettent d'affirmer que :

$$\hat{R}_n(f) \underset{n\to+\infty}{\xrightarrow{p.s}} R_{\mathbb{P}}(f), \qquad \sqrt{n}\{\hat{R}_n(f) - R_{\mathbb{P}}(f)\} \underset{n\to+\infty}{\xrightarrow{\mathcal{L}}} \mathcal{N}(0, Var[l(Y, f(X))]).$$

Ces théorèmes nous permettent ainsi d'avoir un ordre de grandeur de la déviation du risque empirique $\hat{R}_n(f)$ vis-à-vis de sa moyenne $R_{\mathbb{P}}(f)$.

Enfin, nous ne disposons en général que d'une certaine classe $\mathcal{F} \subseteq \mathcal{F}(\mathcal{X}, \mathcal{Y})$ de fonctions candidates afin de minimiser notre risque. Pour éviter les phénomènes de sur-apprentissage qui consiste à apprendre par coeur les sorties Y_i associées aux entrées X_i, il n'est pas judicieux de choisir $\mathcal{F} = \mathcal{F}(\mathcal{X}, \mathcal{Y})$. L'apprentissage revient donc à déterminer une fonction $f^*_{\mathcal{F}}$ de la classe $\mathcal{F}$ qui minimise le risque soit :

$$f^*_{\mathcal{F}} \in \underset{f\in\mathcal{F}}{argmin}\ \mathbb{E}_{\mathbb{P}}[l(Y, f(X))]$$

La fonction $f^*_{\mathcal{F}}$ est bien évidemment un idéal théorique que l'on souhaite atteindre. En pratique, on ne peut accéder qu'à un estimateur $\hat{f}_{n,\mathcal{F}}$ de la fonction $f^*_{\mathcal{F}}$ construit à partir de notre base de données. Le problème d'apprentissage revient donc d'une part déterminer au mieux $\hat{f}_{n,\mathcal{F}}$ de sorte à minimiser $R_{\mathbb{P}}(\hat{f}_{n,\mathcal{F}})$ et d'autre part à minimiser l'écart entre le risque $R_{\mathbb{P}}(\hat{f}_{n,\mathcal{F}})$ et le risque $R_{\mathbb{P}}(f^*)$.

Si l'on regarde ce dernier écart de plus près, on constate que

$$R_{\mathbb{P}}(\hat{f}_{n,\mathcal{F}}) - R_{\mathbb{P}}(f^*) = [R_{\mathbb{P}}(\hat{f}_{n,\mathcal{F}}) - R_{\mathbb{P}}(f^*_{\mathcal{F}})] + [R_{\mathbb{P}}(f^*_{\mathcal{F}}) - R_{\mathbb{P}}(f^*)].$$

Cette expression fait apparaître deux termes. Le premier est appelé erreur stochastique ou erreur d'estimation (pouvant être assimilé à la variance de l'erreur de modèle). Le deuxième représente l'erreur systématique (ou biais). On note que plus la classe $\mathcal{F}$ est importante plus l'erreur systématique est faible. A l'inverse, plus la classe $\mathcal{F}$ est importante plus l'erreur stochastique est grande. En d'autre terme, moins l'erreur est importante sur notre base d'apprentissage, moins l'on a de chance d'obtenir un bon modèle prédictif (c'est-à-dire que l'on posséde une variance élevé). Ce phénomène est appelé sur-apprentissage. Le choix de $\mathcal{F}$ représente donc un axe de recherche et un enjeu cruciale que l'on retrouve dans la littérature statistique [Dalayan 2014] [Cesa-Bianchi and Lugosi 2006].

Il est important de rappeler que dans un cadre d'assurance, l'interprétation des fonctions de la classe $\mathcal{F}$ est nécessaire pour la bonne compréhension des mécanismes mis en place par les compagnies. Dans certains cas, lorsqu'une interprétation du modèle est nécessaire, $\mathcal{F}$ ne peut être une classe contenant que des fonctions non-paramétriques ou alors il convient d'utiliser des méthodes d'interprétation supplémentaires (Cf Chapitre 6).

L'apprentissage statistique apporte une nouvelle approche sur l'estimation de la fonction f introduite dans le paragraphe 2.1.5 page 42. Plus précisément, il offre à l'actuaire un nouveau panel de modèles candidats à l'amélioration du contrôle de ses risques. Nous nous sommes plus particulièrement intéressé à l'utilisation des réseaux de neurones.

2.2 L'INTRODUCTION DE NOUVEAUX MODÈLES EN ASSURANCE

A partir des informations que nous fournissent nos différents capteurs sensoriels, le cerveau humain a la capacité de capter et d'inférer de nombreux effets. Depuis 1950, sur la base de ce schéma, les chercheurs tentent alors de répliquer ce fonctionnement. A l'origine, les limites informatiques ne permettaient que l'élaboration de structures simples. Aujourd'hui la multiplication des capacités de calcul a permis de remettre au goût du jour les réseaux de neurones.

Cependant, il est important de souligner qu'à ce jour, les réseaux de neurones sont des modèles appartenant au domaine de l'apprentissage statistique. A ce titre et contrairement à la discipline de l'intelligence artificielle, les réseaux de neurones ont besoin d'être entraînés à partir d'un historique de données. Ils sont, en outre, pas dans la capacité d'inférer de nouvelles règles. Les analogies au cerveau humain souvent utilisées ne sont qu'illustratives. Nous serons donc vigilants à ne pas "sur-interpréter" les résultats fournis par les réseaux de neurones. Néanmoins, si leur utilisation est d'actualité c'est qu'ils capturent des phénomènes complexes que les statistiques usuelles ne permettent pas toujours d'appréhender.

2.2.1 *L'utilisation des réseaux de neurones artificiels*

2.2.1.1 *Les réseaux de neurones dans la littérature actuarielle*

Dans la littérature actuarielle, les réseaux de neurones ont été très peu utilisés à ce jour. Dans les publications actuarielles recensées qui en font référence avant 2015, leurs études aboutissent à un bon pouvoir prédictif lorsque le modèle est entraîné sur des historiques de données. A ce titre, ils permettent par exemple de prédire la hausse (ou la baisse) du CAC40 avec 80% de précision [Richard 2013] ou encore d'établir une tarification en santé [Aouizerate 2010] plus adaptée que les méthodes classiques. Dans ce dernier cadre, "79% des tarifs prédits par le modèle neuronal se sont avérés être plus proches de la consommation réelle que les taux de cotisations pratiqués issus d'outils de tarification standard" [Aouizerate 2010].

Tester les réseaux de neurones sur des valeurs de portefeuille d'actifs simulées apparaît comme une approche novatrice. En outre, cette étude nous fournira de plus amples informations sur la manière dont on calibre un réseau de neurones, la façon dont il s'adapte

à des données fortement bruitées mais également sur la façon d'implémenter un tel algorithme d'apprentissage statistique.

Enfin utiliser les neurones artificiels à des fins de régression dans un cadre de *Least Square Monte Carlo* nous permettra d'en savoir un peu plus sur ce modèle très prisé aujourd'hui en classification dans l'analyse sémantique et l'analyse d'image par le monde du digital. Nous détaillons donc par la suite le fondement théorique des réseaux de neurones. Ces notions seront approfondies dans le chapitre 4 dédié à l'exploration des réseaux de neurones profonds.

2.2.1.2 *Un outil auquel s'intéressent les GAFAM et de nombreuses industries*

Les géants de l'industrie du digital que sont Google, Amazon, Facebook, Apple et Microsoft (GAFAM) manifestent aujourd'hui un intérêt particulier pour les réseaux de neurones. Ces modèles semblent intéresser les GAFAM par leur bon pouvoir prédictif en analyse sémantique et surtout dans l'analyse d'image. Plus concrètement, leur approche consiste à utiliser des réseaux de neurones profonds[15] (Deep Learning) possédant un nombre de couches cachées élevé pour s'assurer de capter un maximum d'effets non linéaires qu'un humain ne pourrait détecter par une approche classique. Si ces techniques sont possibles de nos jours c'est qu'elles s'implémentent plus facilement grâce aux nouvelles technologies (ordinateurs mutliprocesseur, apparition des *clusters*, etc.).

Si la raison qui motive l'utilisation des neurones dans le monde digital ne présente pas encore aujourd'hui d'intérêt direct pour l'assureur, elle interroge néanmoins sur les capacités prédictives mais également sur la possibilité de capter des effets non linéaires et donc d'en connaître un peu plus sur des fonctions complexes comme celles qui régissent les valeurs actuelles nettes des passifs d'une entreprise en fonction des scénarios économiques ou encore le comportement d'un arbitragiste.

2.2.2 *Présentation d'un neurone artificiel*

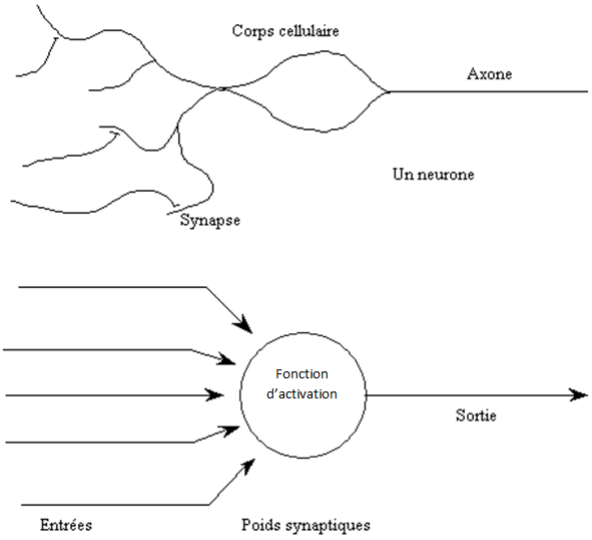
Le premier neurone artificiel fonctionnel fut introduit par Franck Rosenblatt en 1957 [Rosenblatt 1960]. Ce neurone qualifié de nos jours d'élémentaire porte le nom de Perceptron, il a permis dans ses premières utilisations à déterminer le sexe d'un individu présenté au travers d'une photo. Si ce premier neurone est important c'est qu'il introduit le premier formalisme mathématique d'un neurone biologique. On peut donc décrire un neurone artificiel par analogie avec une cellule nerveuse :

15 http://bigbrowser.blog.lemonde.fr/2015/06/23/quand-les-ordinateurs-voient-des-formes-dans-les-nuages/

- les synapses apportant l'information à la cellule sont formalisés par un vecteur réel. La dimension du vecteur d'entrée du neurone (qui n'est autre qu'une fonction) correspond biologiquement au nombre de connections synaptiques;
- chaque signal apporté par un synapse est ensuite analysé par la cellule. Mathématiquement, ce schéma est transcrit par la pondération des différents éléments constitutifs du vecteur d'entrée;
- en fonction de l'information acquise, le neurone décide de retransmettre ou non un signal au travers de l'axome. Ce phénomène est répliqué par l'introduction d'une fonction d'activation. Le signal de sortie est modélisé par un nombre réel calculé comme image par la fonction d'activation du vecteur d'entrée pondéré.

Ainsi, un neurone artificiel est un modèle semi-paramétrique. Le choix de la fonction d'activation est en effet laissé à l'utilisateur. Nous introduisons dans le paragraphe qui suit une formalisation rigoureuse qui nous permettra de poser le modèle.

Figure 9.: Analogie schématique entre un neurone biologique et artificiel



2.2.3 *Introduction du modèle*

Sur la base du schéma présenté dans la partie précédente, on peut définir un neurone élémentaire formellement par :

1. Un espace d'entrée $\mathcal{X}$ généralement $\mathbb{R}^n$ avec $n \in \mathbb{N}^*$;

2. Un espace de sortie $\mathcal{Y}$ généralement $\mathbb{R}$ ou un ensemble dénombrable fini (par exemple $\{-1, 1\}$);

3. Un vecteur de paramètres $w \in \mathbb{R}^k$ où k peut être différent de n (par exemple lorsque l'on souhaite introduire un offset dans le cas d'un Kernel Trick);

4. Une fonction d'activation $\phi : \mathbb{R} \to \mathbb{R}$. Cette fonction doit être dans l'idéal monotone, dérivable et saturante[16] afin de s'assurer de certaines propriétés de convergences d'après le théorème de Cybenko (ressemble aux théorèmes de Stone-Weirstrass et sera présenté plus loin).

et on appellera neurone toute application f_w de $\mathcal{X}$ dans $\mathcal{Y}$ définie par

$$f_w(x) = \phi(w^T x) \quad \forall x \in \mathcal{X}.$$

Voici une liste de fonctions d'activation ϕ définies sur $\mathbb{R}$ couramment utilisées pour les réseaux de neurones :

Table 3.: Exemple de fonctions d'activation ϕ utilisées

Fonction	Définition	Image
Tanh	$x \to \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$[-1, 1]$
Sigmoide[17], $\alpha \in \mathbb{R}^+$	$x \to \frac{1}{1+e^{-\alpha x}}$	$[0, 1]$
Arctan	$x \to arctan(x)$	$]-\frac{\pi}{2}, \frac{\pi}{2}[$

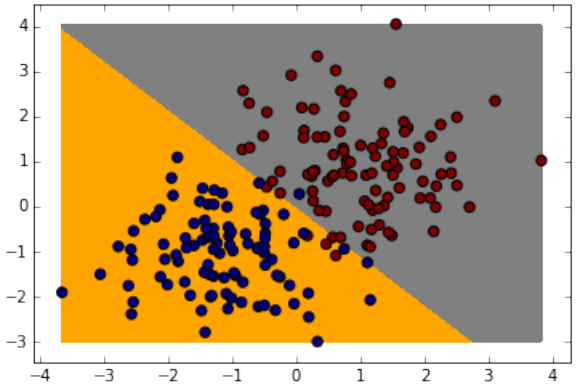
Si l'on revient au perceptron introduit par Rosenblatt [Rosenblatt 1960], on assimile un neurone élémentaire à la fonction :

$$f_w : x \in \mathbb{R}^n \to y = sign(w^T x).$$

Si l'on prend de plus le cas particulier où $n = 2$ et $\mathcal{Y} = \{-1, 1\}$, un Perceptron s'interprète géométriquement comme une droite passant par l'origine. Résoudre un problème d'apprentissage statistique où $\mathcal{Y} = \{-1, 1\}$ et $\mathcal{X} = \mathbb{R}^2$ revient alors à déterminer une droite passant par l'origine qui permet de séparer l'ensemble des points en deux sous-ensembles comme l'illustre la Figure 10. Les points situés au-dessus de la droite sont alors associés au label 1 alors que les autres sont associés au label -1.

16 Nous renvoyons le lecteur à [Cybenko 1989] pour la définition mathématique de ce terme
17 Le terme sigmoide est ici abusif car il désigne en réalité une classe de fonctions plus générale

Figure 10.: Répresentation graphique de la frontière de décision engendré par un Perceptron



On remarque que selon cette formalisation, beaucoup de modèles statistiques plus courants en actuariat pourraient être vus comme des neurones. En effet si l'on regarde d'un peu plus près, tout modèle GLM (Generalized Linear Model) pourrait s'interpréter comme un neurone formel où la fonction d'activation ϕ n'est d'autre que l'inverse de la fonction de lien canonique (par exemple). Soit si g désigne la fonction de lien du GLM, w le vecteur de paramètres, Y la variable à expliquer et X le vecteur des variables explicatives de même dimension que w :

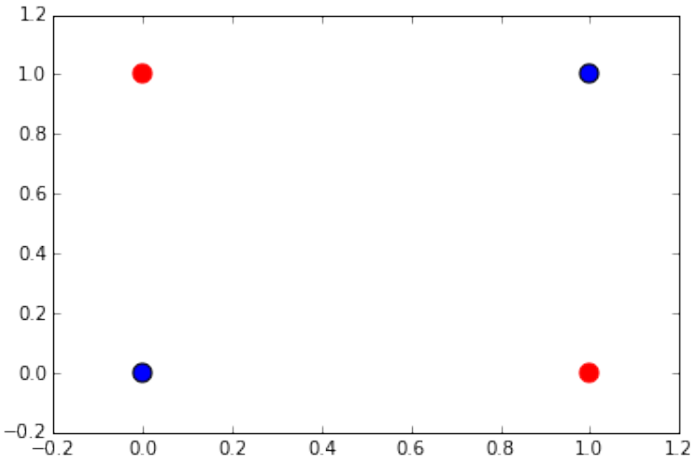
$$g(E[Y|X]) = w^T X.$$

On retrouve la modélisation neuronale en prenant $\phi = g^{-1}$. Cependant, là où réside la différence majeure entre les GLM et le modèle neuronal est que ce dernier n'introduit aucune hypothèse de distribution sur $Y|X$. D'autre part, lorsque le nombre de neurones par couche augmente, la convergence n'est pas nécessairement garantie si la fonction d'activation ne vérifie pas certaines propriétés (qu'on ne retrouve pas dans la majorité des fonctions de liens canoniques des GLM).

2.2.4 *Les réseaux de neurones multicouches*

Le Perceptron présente néanmoins certaines limites qui mènent à l'élaboration de structures plus complexes et donc plus coûteuses en calcul. Pour mieux comprendre les limites du Perceptron linéaire, prenons le cas d'un échantillon de 4 vecteurs de $\mathcal{X} = \mathbb{R}^2$ dont le label associé est choisi dans l'ensemble $\mathcal{Y} = \{-1, 1\}$. Plus précisément, choisissons ces points répartis selon la Figure 11:

Figure 11.: Illustratrion d'un cas non séparable



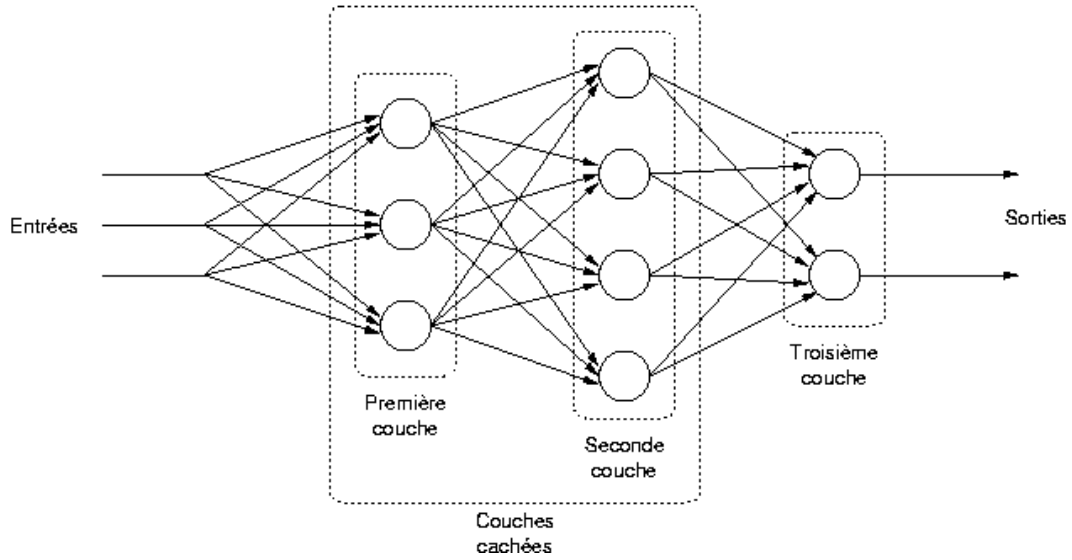
On visualise ici les premières limites des modèles linéaires : il n'existe aucune droite permettant de séparer les points bleus des points rouges. Plus formellement, cela se traduit par la dimension de Vapnik (par exemple exposé dans [Dalayan **(year?)**]) trop faible associée à la classe $\mathcal{F}$ des fonctions considérées. Cette notion complexe n'est pas détaillée dans ce chapitre.

C'est la raison pour laquelle des structures plus complexes sont utilisées afin de capter des effets non linéaires. Un réseau de neurones est ainsi une association de différents neurones élémentaires formels connectés entre eux de façons plus ou moins complexes.

2.2.4.1 *Présentation du modèle multicouches*

Dans ce chapitre nous donc allons considérer des structures de réseaux de neurones multicouches unidirectionnelles. Pour faire l'analogie avec le cerveau humain, on présentera des structures sans rétroaction comme indiqué sur la Figure 12. Le message nerveux partira des capteurs et se transmettra de neurones en neurones sans faire de "retour en arrière" jusqu'à la terminaison nerveuse finale.

Figure 12.: Structure d'un réseau de neurones multicouches considéré dans notre étude



Pour faire le lien avec la formalisation déjà présentée sur la Figure 12 chaque cercle représente un neurone élémentaire. Chaque rectangle englobant plusieurs cercles représente une couche. On parle de couche d'entrée pour la première couche prenant en "input" les observation $x \in \mathcal{X}$, de couche de "sortie" pour la couche fournissant en "output" la prédiction $\hat{y} \in \mathcal{Y}$. Les autres couches sont couramment appelées couches cachées.

De façon plus formelle, chaque couche peut être associée à une fonction de transfert qui prend en entrée un vecteur et fournit en sortie un autre vecteur. Dans le cadre des réseaux de neurones, chaque fonction de transfert est une fonction paramétrique avec un nombre de paramètres proportionnel aux nombres de neurones de chaque couche. On introduit les notations suivantes :

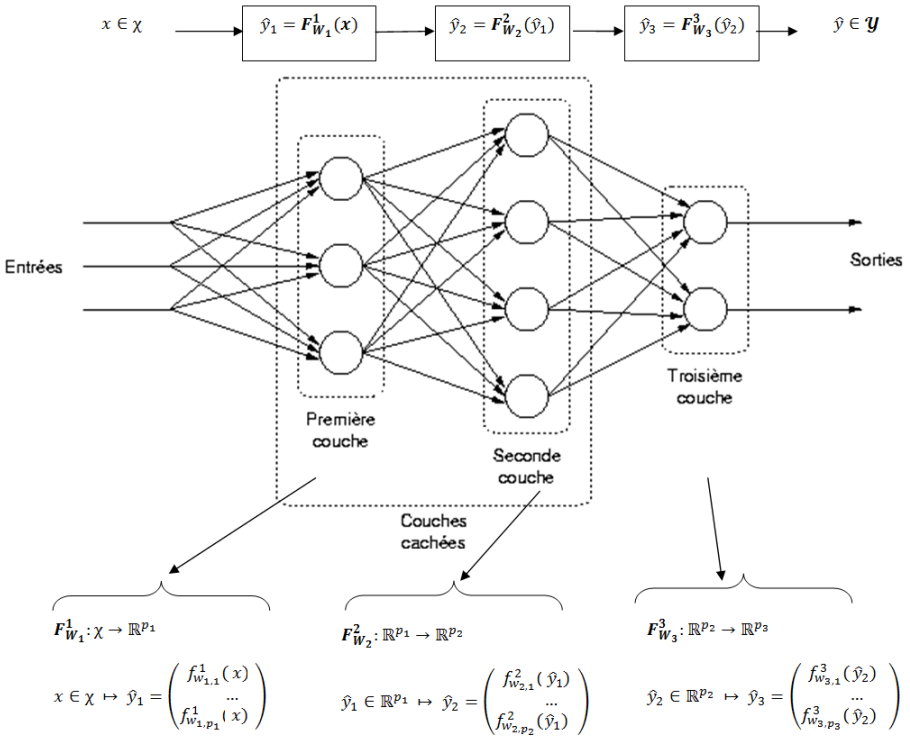
- $K \in \mathbb{N}^*$: nombre de couches ;
- $\forall k \in [[1, K]]$, p_k représente le nombre de neurones dans la couche k ;
- $\forall k \in [[1, K]]$, W_k désigne la matrice des paramètres associée à la couche k. Plus précisément, $W_k \in \mathcal{M}_{p_k, p_{k-1}}(\mathbb{R})$ et pour tout $l \in [[1, p_k]]$, $w_{k,l} \in \mathbb{R}^{p_{k-1}}$ désigne le vecteur de poids associé au neurone élémentaire l de la couche k;
- On appellera $W = \{W_1, .., W_K\}$, l'ensemble des paramètres associés au réseau de neurones ;

- $F^k_{W_k} : \mathbb{R}^{p_{k-1}} \to \mathbb{R}^{p_k}$ désigne la fonction de transfert associée à la couche k. Pour des raisons de simplification, on pourra également écrire F^k ;
- $\hat{y}_k \in \mathbb{R}^{p_k}$ représentera le vecteur image de la couche $k \in [[1, K]]$;
- On appellera $F = F_W = F^1 \circ ... \circ F^K$ la fonction de transfert associée au réseau global. A ce titre, si $x \in \mathcal{X}$, on pourra noter $\hat{y} = F_W(x)$.

Un réseau de neurones multicouches est donc également un modèle semi-paramétrique dont les paramètres sont l'ensemble des composantes des matrices W_k pour tout entier k de $[\![1, K]\!]$. Chaque fonction d'activation associée à chaque neurone (chaque cercle de la Figure 13) est à déterminer par l'utilisateur.

Pour illustrer ces notations, on s'appuiera par le suite sur le schéma de la Figure 13

Figure 13.: Exemple de notations associées aux réseaux de neurones multicouche



2.2.4.2 *Un estimateur de fonctions*

Si cette méthode semble être utile pour approcher des fonctions continues, c'est qu'elle s'appuie sur un théorème fondamental, le théorème de Cybenko [Cybenko 1989] :

Théorème 1. Soit ϕ une fonction réelle continue, non constante, croissante sur $\mathbb{R}$ et discriminante[18] selon la mesure de Lebesgue (toute fonction bornée saturante est discriminante [Cybenko 1989]). Soit $m \in \mathbb{N}*$, soit f une fonction continue sur l'hypercube $I_m = [0,1]^m$ alors :

$\forall \epsilon > 0\ \exists N \in \mathbb{N}^* \mid \forall n > N,\ \exists (w_i)_{i\in[[1,n]]} \in (\mathbb{R}^m)^n, \exists (\alpha_i, b_i)_{i\in[[1,n]]} \in \mathbb{R}^n \times \mathbb{R}^n$ tels que :

$$\forall x \in I_m, \quad \begin{cases} F^n(x) = \sum_{i=1}^n \alpha_i \phi(w_i^T x + b_i) \\ |F^n(x) - f(x)| < \epsilon \end{cases}$$

Ce théorème est par ailleurs appelé théorème d'approximation universelle.

Ce théorème d'analyse permet de justifier la possibilité d'approcher toute fonction continue par un réseau de neurones. C'est la raison pour laquelle ce type de modèle est beaucoup utilisé afin d'approcher un critère de décision inconnu tout en élargissant l'ensemble des fonctions candidates $\mathcal{F}$ (Cf paragraphe 2.1.6.2). Ainsi pour une tolérance d'erreur donnée, il existe alors un réseau de neurones à une couche avec un nombre fini de neurones (même si potentiellement élevé) qui permet d'approcher n'importe quelle fonction de décision continue.

Ce théorème ne fournit cependant aucune information sur la vitesse de convergence des suites de fonction F_n vers la fonction f. Il n'indique d'autre part qu'un choix large de fonctions candidates comme fonction d'activation Φ. Cependant, il est d'usage d'utiliser en général des fonctions continues, monotones et bornées comme fonctions d'activation. En effet, Cybenko [Cybenko 1989] a démontré que toute fonction saturante[19] bornée est discriminante.

On rappelle ainsi les fonctions d'activation couramment utilisées que nous utiliserons également dans ce chapitre. Elles ont notamment pu être recensées dans les différents articles publiés par Google sur la reconnaissance visuelle et sémantique ainsi que dans l'article de Cybenko [Cybenko 1989].

18 Cf [Cybenko 1989]. On peut considérer par simplification qu'une fonction discriminante est une fonction grâce à laquelle on peut dissocier facilement deux états grâce à la valeur que prend cette fonction. La fonction *Logit* est par exemple discriminante.

19 C'est-à-dire admettant une limite finie aux limites de son ensemble de définition. Par exemple, la fonction *Cosinus* n'est pas saturante alors que la fonction $Tanh$ (tangente hyperbolique) l'est. La continuité n'est pas nécessaire pour vérifier cette propriété.

Table 4.: Exemple de fonctions d'activation ϕ utilisées

Fonction Φ	Définition	Ens. Image
Tanh	$x \to \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$[-1, 1]$
Sigmoide, $\alpha \in \mathbb{R}^+$	$x \to \frac{1}{1+e^{-\alpha x}}$	$[0, 1]$
Arctan	$x \to arctan(x)$	$]-\frac{\pi}{2}, \frac{\pi}{2}[$

Du théorème d'approximation universelle de Cybenko, on peut alors en déduire le corollaire suivant que nous démontrons.

Corollaire 2.1. Soit $m \in \mathbb{N}^*$, soit $\mathcal{F}_m(C, \mathbb{R})$, l'ensemble des fonctions continues des sous-ensembles compacts $C \subset \mathbb{R}^m$ dans $\mathbb{R}$. On munit $\mathcal{F}_m(C, \mathbb{R})$ de la norme usuelle $||.||$ des fonctions définies sur un compact. Autrement dit, $\forall f \in \mathcal{F}_m(C, \mathbb{R})$, $||f|| = \underset{x \in C}{sup}|f(x)|$. On note S_m, l'ensemble des fonctions sigmoïdes définies sur $\mathbb{R}^m$ par :

$$S_m = \{\phi_{w,b} | \forall x \in \mathbb{R}^m,\ \phi_{w,b}(x) = \frac{1}{1 + e^{-(w^T x + b)}}, w \in \mathbb{R}^m, b \in \mathbb{R}\}$$

Alors S_m est une base de $\mathcal{F}_m(C, \mathbb{R})$.

Démonstration :

D'après le théorème de Cybenko énoncé précédemment la famille $S_m(w, b)$ est génératrice. Montrons alors que cette famille est libre i.e. montrons que toute sous famille finie est libre :

$$\forall N \in \mathbb{N}^*, \forall (w_i)_{i=1..N} \in (R^m)^N, \forall (b_i)_{i=1..N} \in (\mathbb{R})^N \ tels\ que\ \forall i \neq j, w_i \neq w_j\ ou\ b_i \neq b_j$$

$$\forall (\lambda_i)_{i=1..N} \in (\mathbb{R})^N,\ \sum_{i=1}^{N} \lambda_i \phi_{w_i,b_i} = 0 \Rightarrow \forall i = 1..N,\ \lambda_i = 0$$

Démontrons ce résultat par récurrence.

Pour $N = 1$, le résultat est direct puisque que pour tout $w \in R^m$ et $b \in \mathbb{R}$, il existe au moins un point x de sorte que $\phi_{w,b}(x) \neq 0$ impliquant le résultat pour $N = 1$.

Soit $N > 1$. Supposons l'hypothèse de liberté vérifiée pour toute sous famille de N éléments de S_m Soient $(w_i)_{i=1..N+1} \in (\mathbb{R}^m)^{N+1}$ et $(b_i)_{i=1..N+1} \in (\mathbb{R})^{N+1}$. Soit $(\lambda_i)_{i=1..N+1} \in \mathbb{R}^{N+1}$ tel que $\sum_{i=1}^{N+1} \lambda_i \phi_{w_i,b_i} = 0$.

S'il existe $i \in \{1, .., N+1\}$ tel que $w_i = 0$ i.e. ϕ_{w_i,b_i} constante, l'hypothèse à $N+1$ est alors vérifiée. Sinon, supposons que w_i est non nul. Ainsi, soit $(x_i)_{i=1,..,N+1} \in (\mathbb{R}^m)^{N+1}$. On a

$$\forall i = 1, .., N+1 \ \sum_{i=1}^{N+1} \lambda_j \phi_{w_j,b_j}(x_i) = 0 \quad (S)$$

Ce système d'équation, si on note $\Phi_{i,j} = \phi_{w_j,b_j}(x_i)$ peut encore s'écrire à l'aide d'une itération de la méthode du pivot de Gauss.

$$\left\{ \begin{array}{lllll} \lambda_1\Phi_{1,1} & + & \lambda_2\Phi_{1,2} & +...+ & \lambda_{N+1}\Phi_{1,N+1} = 0 \\ 0 & + & \lambda_2(\Phi_{2,2}\Phi_{1,1} - \Phi_{1,2}\Phi_{2,1}) & +...+ & \lambda_{N+1}(\Phi_{2,N+1}\Phi_{1,1} - \Phi_{1,N+1}\Phi_{2,1}) = 0 \\ ... & & & & \\ 0 & + & \lambda_2(\Phi_{N+1,2}\Phi_{1,1} - \Phi_{1,2}\Phi_{N+1,1}) & +...+ & \lambda_{N+1}(\Phi_{N+1,N+1}\Phi_{1,1} - \Phi_{1,N+1}\Phi_{N+1,1}) = 0 \end{array} \right.$$

Notons $\Lambda_* = (\lambda_i)_{i=2,.,.N+1}$ et Δ_*,M_*:

$$\begin{array}{rcl} M_* & = & (\Phi_{i,j})_{i,j=2,..,N+1} \\ \Delta_* & = & (\Phi_{1,j}\Phi_{i,1})_{i,j=2,..,N+1} \end{array} .$$

Donc (S) est équivalent à :

$$\left\{ \begin{array}{ccccccc} \lambda_1\Phi_{1,1} & + & \lambda_2\Phi_{1,2} & + & ... & + & \lambda_{N+1}\Phi_{1,N+1} = 0 \\ 0 & + & (\Phi_{1,1}M_* - \Delta_*)\Lambda_* & & & & = 0 \end{array} \right.$$

Cette équation est par ailleurs valable pour tout $(x_i)_{i=1..N+1} \in (\mathbb{R}^m)^{N+1}$. En particulier, si on pose $x_1 = ax_1$ avec a un coefficient réel, on peut alors remarquer que par définition des fonctions sigmoïdes, il existe $(\alpha_i)_{i=1,..,N+1} \in \{0,1\}^{N+1}$ de sorte que $\alpha_1 = 1$ et :

$$\forall j = 1, .., N+1, \ \lim_{a \to +\infty} \Phi_{i,j}(a) = \lim_{a \to +\infty} \phi_{w_j,b_j}(ax_1) = \alpha_j$$

On définit :

$$\begin{array}{rcl} U_* & = & (\Phi_{2,1}, ..., \Phi_{N+1,1})^T \\ V_* & = & (\alpha_2\Phi_{2,1}, ..., \alpha_N\Phi_{N+1,1})^T \\ L_* & = & (V_*)_{i=2..N+1} \text{ dont les N colonnes sont identiques} \end{array} .$$

On remarque par ailleurs, qu'en remplaçant x_1 par ax_1 dans le système (S), seules les termes dépendant de $\Phi_{1,j}$ dépendent alors de a. Par la suite on fera apparaître cette dépendance en notant $*(a)$ avec $*$ l'expression concernée par la dépendance. Ainsi on peut vérifier que :

$$\lim_{a\to+\infty} \Delta_*(a) = L_*$$

(S) équivaut donc à

$$\begin{cases} \lambda_1\Phi_{1,1}(a) + \lambda_2\Phi_{1,2}(a) + ... + \lambda_{N+1}\Phi_{1,N+1}(a) = 0 \\ 0 + [\Phi_{1,1}(a)M_* - (L_* + (\Delta_*(a) - L_*))]\Lambda_* = 0 \end{cases}$$

Ce qui est encore équivalent à :

$$\begin{cases} \lambda_1\Phi_{1,1}(a) + \lambda_2\Phi_{1,2}(a) + ... + \lambda_{N+1}\Phi_{1,N+1}(a) = 0 \\ 0 + (\Phi_{1,1}(a)M_* - L_*)\Lambda_* + (\Delta_*(a) - L_*)\Lambda_* = 0 \end{cases}$$

En considérant la limite $a \to +\infty$ et la définition de L_* dans la seconde équation de (S), on a :

$$\begin{array}{lcll} & (\Delta_*(a) - L_*) & \underset{a\to+\infty}{\to} & 0 \\ \Rightarrow & (\Phi_{1,1}(a)M_* - L_*)\Lambda_* & \underset{a\to+\infty}{\to} & 0 \\ \Rightarrow & (M_* - L_*)\Lambda_* & = & 0 \text{ car } \Phi_{1,1}(a) \text{ tend vers 1 par définition} \\ \Leftrightarrow & M_*\Lambda_* - (\sum_{j=2}^{N+1}\lambda_j)V_* & = & 0. \end{array}$$

Donc $M_*\Lambda_* - (\sum_{j=2}^{N+1}\lambda_j)V_* = 0$. Or d'aprés l'hypothèse de récurrence on a $\lambda_j = 0$ pour tout $j = 2, .., N+1$. En utilisant par ailleurs que $\lim_{a\to+\infty} \lambda_1\Phi_{1,1}(a) = \lambda_1$ par (S) on obtient $\lambda_1 = 0$

Ainsi, ce corollaire confirme le fait qu'utiliser un réseau de neurones (avec des fonctions d'activation sigmoïde) permet d'approcher une fonction continue.

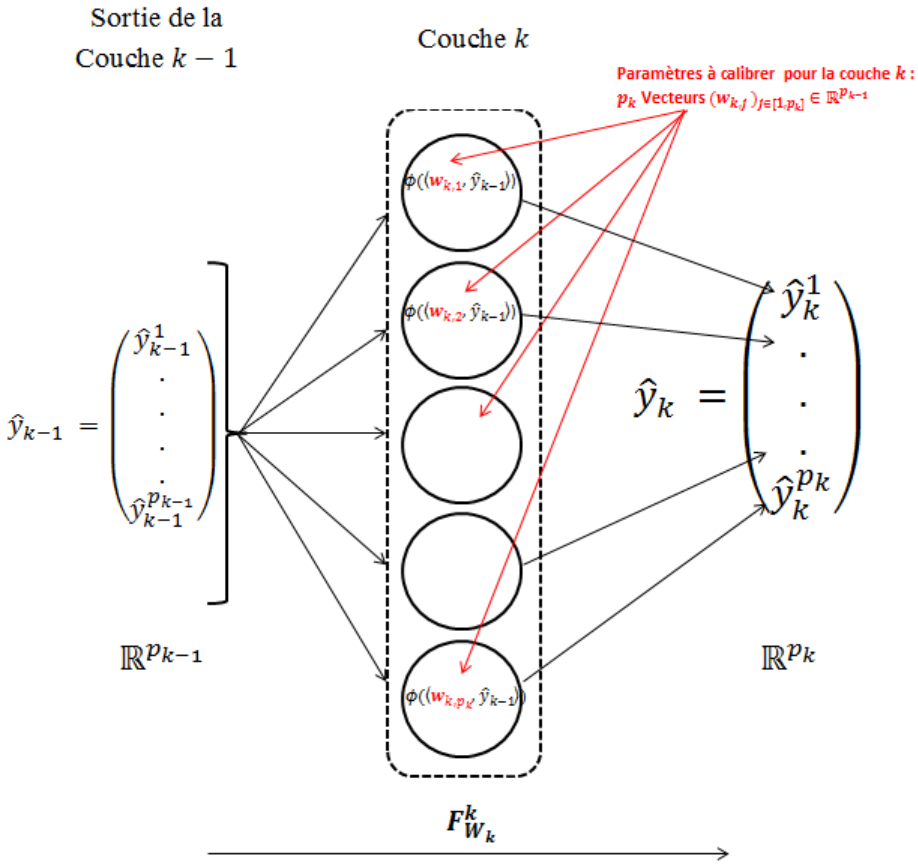
2.2.5 *L'apprentissage du modèle*

Si l'on se rappelle du schéma de la Figure 13 précédent, dans chaque couche $k \in [\![1, K]\!]$ existe p_k neurones élémentaires . Chaque cercle représente une fonction de type :

$$f : x \in \mathbb{R}^{p_{k-1}} \to \phi(w^T.x) \in \mathbb{R}.$$

Ainsi, pour la couche k , on doit calibrer autant de vecteurs $w \in \mathbb{R}^{p_{k-1}}$ que de neurones présents dans la couche, c'est-à-dire p_k. On peut ainsi paramétrer chaque couche par une matrice de paramètres $W_k \in \mathcal{M}_{p_{k-1},p_k}(\mathbb{R})$. On prendra comme convention que $p_0 = dim(\mathcal{X})$ c'est-à-dire la dimension de l'espace d'entrée.

Figure 14.: Illustration des paramètres à calibrer pour une couche



Dans cette partie nous allons donc voir étape par étape, une manière de calibrer un modèle d'apprentissage statistique : les réseaux de neurones. Ces différentes étapes apparaissent régulièrement en apprentissage supervisé et nous verrons dans notre étude que certaines d'entre elles ont été adaptées pour une implémentation informatique plus aisée.

Dans la suite, nous considérons que les fonctions d'activation sont fixées *a priori*. Après avoir présenté la méthode permettant de calibrer les différents paramètres, nous discuterons le choix des fonctions d'activation ϕ.

2.2.5.1 *Le choix de la mesure de risque pour le calibrage des paramètres*

Une fois les paramètres à calibrer du modèle sont identifiés (ici les réels constituant les matrices W_k pour chaque couche $k \in [\![1, K]\!]$), il est nécessaire de fixer une fonction de perte l. En effet, on rappelle que l'objectif de l'apprentissage supervisé sur une base

d'apprentissage de $n \in \mathbb{N}^*$ couples $(y_i, X_i) \in \mathcal{Y} \times \mathcal{X}$ est de minimiser le risque empirique :

$$\hat{R}_n(F_W) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, F_W(X_i)).$$

Pour une structure de réseau de neurones fixée (c'est-à-dire nombre de couches, nombre de neurones par couche et fonctions d'activation fixés), le programme revient donc à déterminer l'ensemble de paramètres $W^* = (W_1, ..., W_K)$ de sorte que :

$$W^* \in arg \min_{W=(W_1,...,W_K)} \frac{1}{n} \sum_{i=1}^{n} l(y_i, F_W(X_i)).$$

De cette formule apparaît l'importance du choix de la fonction l. Appelée fonction de perte, elle quantifie l'erreur moyenne commise par notre modèle F_W sur la base d'apprentissage. *A priori* l peut être choisie arbitrairement. Cependant, dans l'optique de résoudre un programme d'optimisation, on préférera des fonctions de coût sous-différentiables et convexes afin de garantir la convergence des algorithmes d'optimisation. Le tableau ci-dessous présente quelques exemples de fonctions de perte :

Table 5.: Exemple de fonctions de perte utilisées, y représente la "vraie" valeur et $\hat{y}$ la valeur prédite par le modèle

Fonction	Définition
Hinge	$\hat{y} \to max(0, 1 - y\hat{y})$
SquareLoss	$\hat{y} \to \|\|y - \hat{y}\|\|_2$
Logistic	$\hat{y} \to ln(1 - e^{-y\hat{y}})$

Cette liste est loin d'être exhaustive et peut être complétée selon les contraintes de l'utilisateur.

Par la suite, nous allons dans un premier temps utiliser les réseaux de neurones à des fins de régression. Nous souhaitons comparer nos résultats à ceux obtenus par régression linéaire usuelle. Nous travaillerons avec la mesure *SquareLoss*, c'est-à-dire la fonction de perte quadratique couramment utilisée en statistique.

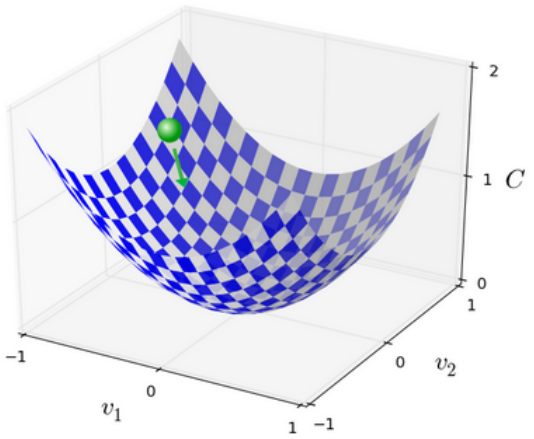
2.2.5.2 *Optimisation des paramètres par méthodes de gradient*

Nous venons donc de poser notre programme d'optimisation à savoir déterminer le jeu de paramètres W permettant de minimiser le risque empirique. Résoudre ce programme revient donc à calibrer notre réseau de neurones.

Une famille de méthodes très utilisée en apprentissage statistique pour résoudre un programme d'optimisation est la descente de gradient. Cette méthode garantit la convergence vers un optimum lorsque la fonction à minimiser est convexe. Elle nécessite d'autre part la sous-différentiabilité du risque ce qui impose des contraintes sur la fonction de coût que nous utilisons.

Plus précisément, les méthodes de descente de gradient sont des algorithmes itératifs qui permettent de converger vers un optimum (pas nécessairement global). Comme l'illustre la Figure 15, partant d'une initialisation aléatoire des paramètres, d'itération en itération, l'algorithme modifie les paramètres de sorte à minimiser le risque.

Figure 15.: Illustration graphique de la méthode de descente de gradient et du déplacement vers un optimum dans un cas simple où le risque C dépend de deux paramètres v_1 et v_2



Il existe plusieurs variantes de méthode de descente de gradient qui permettent de s'adapter à la volumétrie des données mais également la parallélisation des calculs. Nous avons implémenté pour notre modèle neuronal trois méthodes différentes. On rappelle que les paramètres à optimiser sont les différentes composantes des matrices W_k pour $k \in [\![1, K]\!]$. De ce fait, on notera $w^t_{k,(m,n)}$ la valeur du paramètre de la matrice W^t_k, situé à la ligne m et la colonne n pour l'itération t. Nous noterons γ_t la valeur de pas de contrôle du gradient à l'itération t.

La descente de gradient standard *Batch*

Cette méthode est la méthode la plus connue et se rapproche de l'algorithme d'optimisation de Newton-Raphson. Concrètement, à chaque itération t de l'algorithme et ce jusqu'à

ce qu'un certain critère d'arrêt (à déterminer par l'utilisateur), pour chaque paramètre réel $w_{k,(m,n)}$ du modèle à calibrer, on effectue une correction moyenne de sorte à ce que l'ensemble des prédictions $(\hat{y}_i)$ effectuées en observant l'ensemble des (x_i) soit le plus proche possible des vraies valeurs (y_i). Pour chaque itération, on a alors :

$$w_{k,(m,n)}^{t+1} \leftarrow w_{k,(m,n)}^{t} - \gamma_t \frac{\partial \hat{R}_n(F_{W^t})}{\partial w_{k,(m,n)}}$$

ce qui se réécrit :

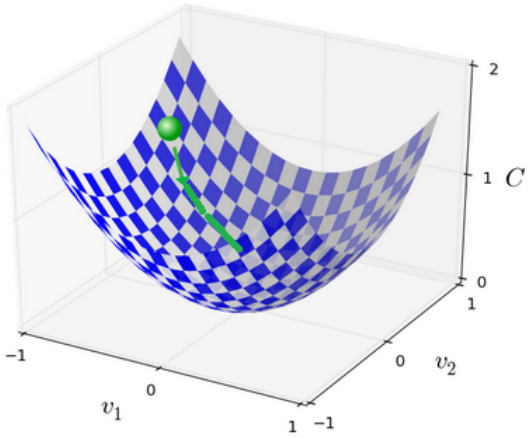
$$w_{k,(m,n)}^{t+1} \leftarrow w_{k,(m,n)}^{t} - \gamma_t \frac{1}{n} \sum_{i=1}^{n} \frac{\partial l(y_i, F_{W^t}(X_i))}{\partial w_{k,(m,n)}}$$

$\forall k \in [[1, K]]$ (K : nombre de couches),
$\forall (m, n) \in \mathbb{R}^{p_{k-1}} \times \mathbb{R}^{p_k}$

Qualitativement, cette méthode permet à chaque itération de tenir compte de l'erreur moyenne commise sur la base d'apprentissage par notre modèle et de corriger les paramètres en conséquence. Comme l'illustre la Figure 16 la convergence vers l'optimum est directe. Néanmoins, cette descente de gradient demande de calculer à chaque itération et pour chaque paramètre, autant de dérivées partielles que d'observations pour effectuer une descente moyenne. On comprend alors que dans le cas de données volumineuses (supérieures à 1 000 000 de lignes), cet algorithme sera peu adapté, à moins d'effectuer une bonne parallélisation des calculs[20].

20 il faudrait que le temps de communication entre les machines ne rende pas l'algorithme plus long que s'il avait été réalisé sur une seule machine

Figure 16.: Illustration graphique de la méthode de descente de gradient *Batch* et du déplacement vers un optimum dans un cas simple où le risque C dépend de deux paramètres v_1 et v_2



La descente de gradient stochastique

L'algorithme de descente de gradient stochastique permet de s'affranchir à chaque itération du calcul du gradient pour chaque observation de notre base d'apprentissage. Plutôt que d'effectuer une descente moyenne à chaque itération, on commence par tirer (sans remise) une observation X_i parmi les n disponibles. On corrige ensuite les paramètres du modèle de sorte à ce que la prédiction faite à partir de X_i soit la plus proche possible de la vraie valeur y_i. On réitère ensuite la méthode jusqu'à avoir parcouru l'ensemble des données. Dans cet algorithme il y a donc autant d'itérations que que d'observations. Contrairement à l'algorithme *batch* à chaque itération un seul vecteur de gradient est calculé (et non plus n). Il est néanmoins parfois nécessaire d'exécuter cet algorithme plusieurs fois pour augmenter la convergence des paramètres du modèle.

On obtient ainsi pour cet algorithme (l'ensemble du procédé peut éventuellement être répété plusieurs fois) :

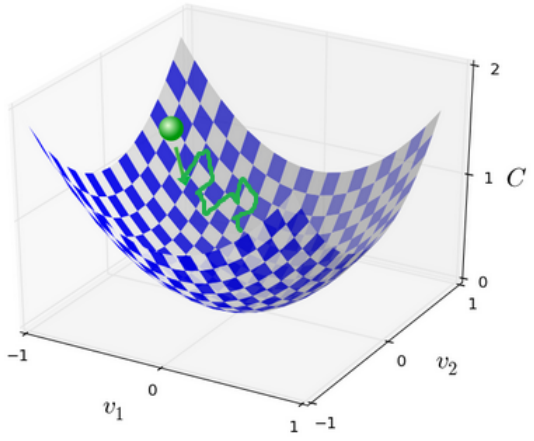
Mélange des données
Pour $t := 1, ..., n$ (il y autant d'itérations que de nombre de lignes dans cet algorithme)
On tire $i \in [\![1, n]\!]$ sans remise.

$$w^{t+1}_{k,(m,n)} \leftarrow w^{t}_{k,(m,n)} - \gamma_t \frac{\partial l(y_i, F_{W^t}(X_i))}{\partial w_{k,(m,n)}}$$

$\forall k \in [[1, K]]$ (K : nombre de couches), $\forall (m, n) \in \mathbb{R}^{p_{k-1}} \times \mathbb{R}^{p_k}$

Cet algorithme peut être réitéré plusieurs fois dans son ensemble selon le besoin de l'utilisateur. L'avantage de cette méthode est qu'à chaque itération, il n'est pas nécessaire de calculer le gradient sur toutes les observations (plus de somme). Elle est donc adaptée aux base de données volumineuses. Cet algorithme s'appuie sur une convergence en probabilité vers un voisinage de l'optimum (et non pas l'optimum lui même). La trajectoire de convergence des paramètres par cette méthode est illustrée par la Figure 17. En réduisant le temps de calcul, on autorise une marge d'erreur plus grande que l'algorithme de descente *batch*. Cette erreur n'est pas nécessairement mauvaise dans la mesure où l'on cherche à éviter les phénomènes de sur-apprentissage.

Figure 17.: Illustration graphique de la méthode de descente de gradient Stochastique et du déplacement vers un optimum dans un cas simple où le risque C dépend de deux paramètres v_1 et v_2



La descente de gradient *mini-batch*

Ce dernier algorithme peut être considéré comme un compromis entre les deux premiers. Plutôt que de considérer à chaque itération l'ensemble des observations (*batch*) ou à l'opposé, une seule observation (stochastique), à chaque itération on considère $b \in \mathbb{N}^*$ observations. b est généralement fixé arbitrairement entre 2 et 100. Ainsi si par exemple $b = 10$ et $n = 1000$ l'algorithme se traduirait par :

Pour $t := 1, 11, 21, 31..., 1000$

$$w^{t+1}_{k,(m,n)} \leftarrow w^{t}_{k,(m,n)} - \gamma_t \frac{1}{b} \sum_{z=1}^{b=10} \frac{\partial l(y_z, F_{W^t}(X_z))}{\partial w_{k,(m,n)}}$$

$\forall k \in [[1, K]]$ (K :nombre de couches), $\forall (m, n) \in \mathbb{R}^{p_{k-1}} \times \mathbb{R}^{p_k}$

Ce dernier algorithme peut présenter de meilleures performances que l'algorithme de descente stochastique si une bonne parallélisation est effectuée. Il s'est avéré néanmoins dans notre étude que l'algorithme stochastique fut le meilleur compromis entre temps de calcul et convergence des paramètres[21].

On voit donc au travers de ces trois algorithmes que plusieurs approches sont alors possibles pour résoudre notre programme d'optimisation. Il existe bien évidemment d'autres méthodes mais nous avons fait le choix de travailler avec celle présentées.

Le choix de la méthode et des méta-paramètres

Parmi les trois méthodes présentées, la méthode la plus satisfaisante en terme de temps de calcul et de convergence fut l'algorithme de gradient stochastique. Nous avons donc favorisé cette méthode. Nous présenterons dans les parties suivantes les résultats obtenus en termes de précision et de temps de calibrage.

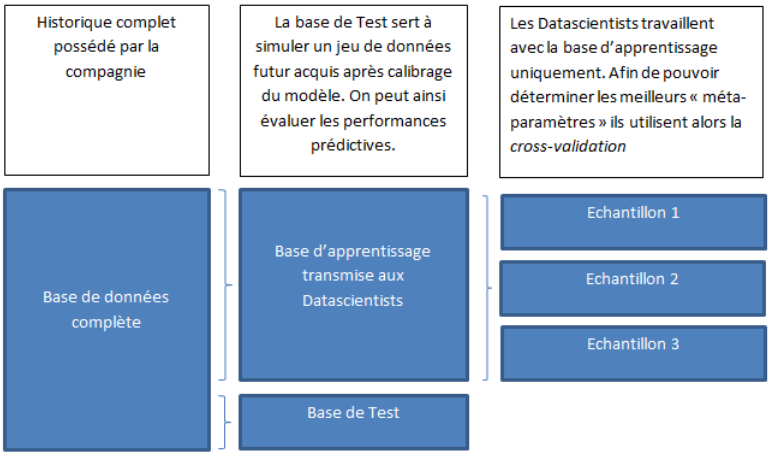
Une fois la méthode de descente de gradient choisie, il reste néanmoins deux paramètres à fixer par l'utilisateur : le nombre d'itérations et le pas de gradient γ_t pour chaque itération t .

Le nombre d'itérations peut être déterminé par la méthode de cross-validation. Avant d'introduire cette méthode il est important de rappeler que le modèle à calibrer est utilisé à des fins de prédictions. De ce fait, afin d'avoir un regard sur les performances prédictives du modèle, nous ne le calibrons pas sur l'intégralité de la base de données dont on dispose mais sur une partie appelée base d'apprentissage (Cf Figure 18). L'autre échantillon sert alors à simuler une base de données acquise dans le futur et ainsi pouvoir comparer les prédictions du modèle aux réalisations.

Pour déterminer les méta-paramètres, la méthode de *cross-validation* est appliquée à la base d'apprentissage. Cette dernière est premièrement échantillonnée. Le modèle est ensuite entraîné sur une partie des échantillons (la base d'entraînement) et testé sur les échantillons restant (la base de validation). En entraînant, puis validant le modèle sur différents échantillons, on obtient ainsi une performance moyenne d'un méta-paramètre. On détermine le méta-paramètre qui fournit en moyenne la meilleur performance de

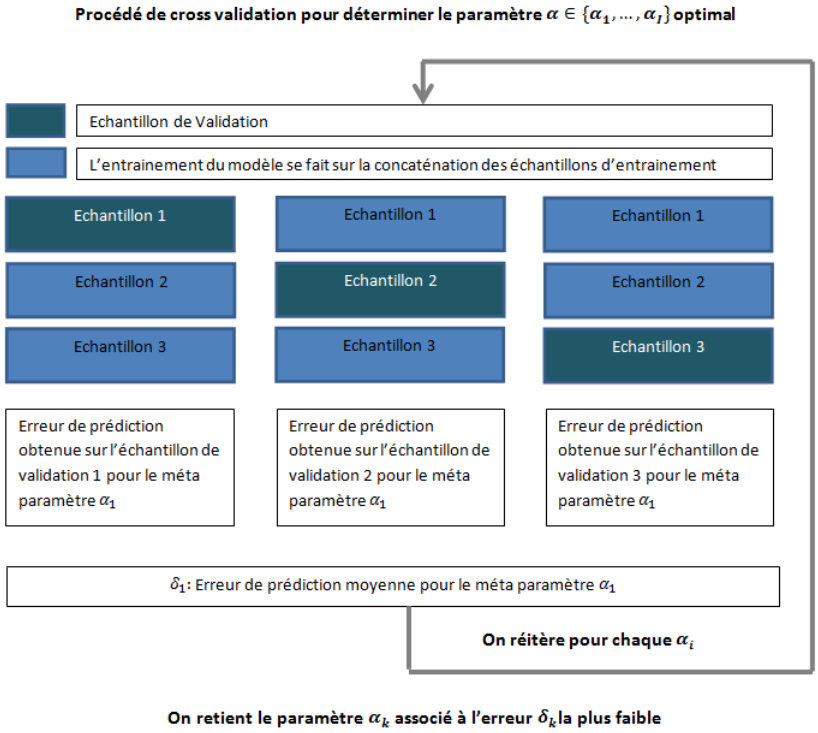
21 On note néanmoins que nos recherches bibliographiques n'ont pas permis de trouver de preuve mathématique quant à la convergence de cette méthode

Figure 18.: Découpage de la base de données pour calibrer et tester le modèle



prédiction en réitérant la manipulation pour différents méta-paramètres. Le schéma de la Figure 19 illustre le procédé.

Figure 19.: Procédé de *cross-validation* afin d'optimiser un paramètre *alpha* (pouvant être par exemple un nombre d'itération, un nombre de neurones, ou un quelconque méta-paramètre)



Ce protocole est essentiel car il permet de simuler différentes distributions des données d'apprentissage et ainsi de rendre le modèle final plus robuste. Enfin, le choix du pas de gradient γ_t est également primordial pour le calibrage. Si ce dernier est trop petit, la correction à chaque itération devient alors trop faible et l'algorithme converge lentement. A l'opposé, si ce dernier est trop important, on risque de dépasser l'optimum et l'algorithme diverge. Ainsi, la littérature [Duchi et al. 2011] propose différentes solutions du choix du pas de gradient. La moins coûteuse en temps de calcul est celle du pas constant. Quelle que soit l'itération t, le pas est identique et fixé par l'utilisateur. Ce fut notre première approche d'implémentation. Seulement, nous avons pu constater qu'à l'approche de l'optimum, l'algorithme de descente de gradient dépassait souvent le point souhaité et divergeait. Afin d'affiner la correction au fur et à mesure de la descente de gradient, nous avons alors introduit une atténuation linéaire de la correction c'est-à-dire que pour tout itération t, $\gamma_t = \frac{1}{1+t}$. Néanmoins, cette atténuation n'est toujours pas suffisante ce qui nous a amené à utiliser (et implémenter) un pas de gradient adaptatif évoluant en fonction de l'itération mais également de la valeur du gradient.

2.2.5.3 *Méthode de gradient adaptative*

Cette méthode est connue sous le nom de AdaGrad (Adaptative Gradient) et a été développée par John Duchi et al. [Duchi et al. 2011]. Elle s'applique plus particulièrement aux algorithmes de descente de gradient stochastique et plus généralement aux méthodes de descente proximale (que nous ne détaillerons pas ici). On rappelle l'algorithme de descente de gradient stochastique :

Mélange des données (ré-indéxation des couples (y_i, X_i))
Pour $t := 1, ..., n$
Pour tout $i \in [\![1, n]\!]$ tiré sans remise.

$$w_{k,(m,n)}^{t+1} \leftarrow w_{k,(m,n)}^{t} - \gamma_t \frac{\partial l(y_i, F_{W^t}(X_i))}{\partial w_{k,(m,n)}}$$

$\forall k \in [[1, K]]$ (K : nombre de couches), $\forall (m, n) \in \mathbb{R}^{p_{k-1}} \times \mathbb{R}^{p_k}$

En outre, J.Duchi définit pour chaque itération t :

$$\gamma_t = \frac{\eta}{\sqrt{\sum_{t'=1}^{t} g_{t'}^2}}$$

Avec pour tout $t' \in [\![1, t]\!]$,

$$g_t' = \frac{\partial l(y_i, F_{W^{t'}}(X_i))}{\partial w_{k,(m,n)}}$$

et η une constante fixée arbitrairement. η représente l'atténuation du gradient à la première itération. Dans certains cas, il est assez judicieux de la choisir relativement faible afin de ne pas diverger dès la première itération.

En d'autres termes pour chaque paramètre, le pas de gradient γ_t (que nous devrions rigoureusement noter $\gamma_{t,k,(m,n)}$) normalise la descente de gradient par la norme euclidienne du vecteur contenant les précédentes valeurs du gradient associées au paramètre $w_{k,(m,n)}$. Ainsi, si la descente de gradient aux itérations précédentes était importante, elle sera atténuée à l'itération suivante. Ce procédé permet ainsi d'éviter le dépassement de l'optimum et de garantir une meilleur convergence [22]. D'autre part, ce procédé ne nécessite que peu de calculs supplémentaires. A chaque itération, que nous employions un pas adapté ou non, le calcul de la dérivée partielle est effectuée. La méthode rajoute simplement une somme de carrés nécessitant peu de temps.

22 L'article de J.Duchi [Duchi et al. 2011] propose une démonstration quant à la convergence vers l'optimal sous des hypothèses de convexité et de différentiabilité

Il existe bien entendu de nombreuses autres méthodes d'optimisation ainsi que d'autres variantes de descente de gradient. De nos jours, la méthode de descente de gradient stochastique est celle qui est le plus souvent utilisée dans le calibrage des réseaux de neurones profonds. Plus particulièrement, la méthode souvent utilisée par défaut est celle proposée par Diederik P. Kingma et Jimmy Lei Ba [Kingma and Lei Ba 2015] et est implémentée dans le plupart des librairies *open source* de *deep learning* comme Keras.

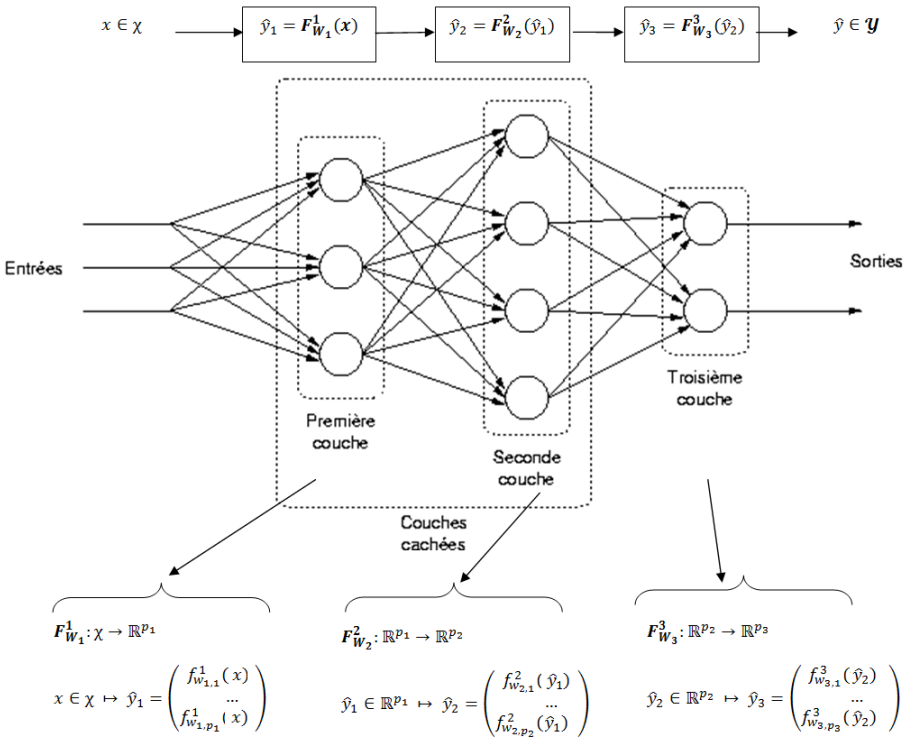
2.2.5.4 *Vers un développement modulaire des réseaux de neurones*

Quel que soit l'algorithme de descente de gradient que l'on utilise, on constate qu'il est nécessaire de pourvoir calculer le gradient de la fonction de perte selon les différents paramètres $w_{k,(m,n)}$ à optimiser. Dans une optique d'implémenter un réseau de neurones, on souhaiterait néanmoins s'affranchir d'un maximum de calculs. En effet, on rappelle, que le nombre de couches d'un réseau ainsi que le nombre de neurones par couche sont autant de méta-paramètres que l'utilisateur doit déterminer. Chaque nouvelle structure de réseau de neurones représente une nouvelle fonction de transfert F_W. Il serait donc assez fastidieux pour l'ajout d'un simple neurone supplémentaire de devoir ré-implémenter un nouveau code.

Pour faire face à ce problème, il suffit d'exploiter pleinement le théorème de dérivation des fonctions composées ainsi que le principe de *backpropagation* que l'on introduit ci-après. Avant cela, on rappelle que chaque couche possède des neurones identiques [23]. Le détail des calculs qui va suivre nous a permis d'implémenter un réseau de neurones modulaire. Nous illustrons la méthode au travers d'un exemple concret. On considére effectivement le réseau de neurones déjà introduit par le schéma précédent de la Figure 13.

23 C'est-à-dire que chaque fonction d'activation est identique au sein d'un même couche

Figure 20.: Rappel : Notations associées aux réseaux de neurones multicouche considérés dans notre étude



Nous considérons donc un réseau de neurones à trois couches :

- L'espace d'entrée $\mathcal{X}$ sera supposé être égal à $\mathbb{R}^{10}$;
- Le réseau posséde deux couches cachées : la première contient $p_1 = 3$ neurones et la deuxième contient $p_2 = 4$ neurones;
- La dernière couche contient $p_3 = 2$ neurones d'agrégation;
- L'espace de sortie $\mathcal{Y}$ sera supposé être égal à $\mathbb{R}^2$.

Selon cet exemple, notre modèle neuronal peut donc se transcrire par la fonction $F_W = F_{W_3} \circ F_{W_2} \circ F_{W_1}$. Notre objectif est alors de calibrer les différents paramètres constituant les matrices W_1, W_2, W_3. Choisissons comme fonction de coût, la fonction convexe et différentiable *SquareLoss* qui est définie par :

$$\forall i \in [\![1, n]\!],\ l(y_i, F_W(X_i)) = ||y_i - F_W(X_i)||_2^2.$$

Le risque empirique $\hat{R}_n(F_W)$ s'exprime :

$$\hat{R}_n(F_W) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, F_W(X_i))) = \frac{1}{n}\sum_{i=1}^{n} ||y_i - F_W(X_i)||_2^2 \ ].$$

Le problème d'optimisation s'écrit donc :

$$\min_W [\ \hat{R}_n(F_W)],$$

avec $W = \{W_1, W_2, W_3\}$ ce qui représente $10 \times 3 + 3 \times 4 + 4 \times 2 = 50$ paramètres à calibrer et pour se faire nous devons évaluer :

$$\frac{\partial l(y_i, F_W(X_i))}{\partial w_{k,(m,n)}}$$

$\forall k \in [[1, K]]$ (K = 3 dans notre exemple), $\forall (m,n) \in \mathbb{R}^{p_{k-1}} \times \mathbb{R}^{p_k}$

Qualitativement, ce terme représente la sensibilité de la fonction de coût au paramètre $w_{k,(m,n)}$. De plus, la couche de sortie est de dimension 2 (on rappelle que dans notre exemple $F_W : \mathbb{R}^{10} \to \mathbb{R}^2$) ainsi pour $X \in \mathbb{R}^{10}$, $F_W(X) = (F_W^{(1)}(X), F_W^{(2)}(X)))$ ce qui donne pour tout $i \in [\![1, n]\!]$:

$$l(y_i, F_W(X_i)) = (y_i^{(1)} - F_W^{(1)}(X_i))^2 + (y_i^{(2)} - F_W^{(2)}(X_i))^2.$$

On en déduit :

$$\frac{\partial l(y_i, F_W(X_i))}{\partial w_{k,(m,n)}} = -2\frac{\partial F_W^{(1)}}{\partial w_{k,(m,n)}}(X_i)(y_i^{(1)} - F_W^{(1)}(X_i)) - 2\frac{\partial F_W^{(2)}}{\partial w_{k,(m,n)}}(X_i)(y_i^{(2)} - F_W^{(2)}(X_i)).$$

On ne détaillera par la suite les calculs que pour une seule des composantes de la couche de sortie (le calcul étant symétrique). Supposons que par exemple $k = 2$, par théorème de dérivation des fonctions composées [24] (en rappelant que $F_W = F_{W_3} \circ F_{W_2} \circ F_{W_1}$) :

$$\frac{\partial F_W^{(1)}}{\partial w_{k=2,(m,n)}}(X_i) = \sum_{q=1}^{p_2=4} \frac{\partial_1 F_{W_3}}{\partial e_q}(F_{W_2} \circ F_{W_1}(X_i)) \times \frac{\partial_q F_{W_2} \circ F_{W_1}}{\partial w_{k=2,(m,n)}}(X_i).$$

Si l'on analyse cette dernière expression, on remarque que le premier terme de la somme représente finalement la dérivée du premier neurone de la troisième couche en fonction de ses coordonnées d'entrée. C'est un terme où les différents paramètres $w_{2,(m,n)}$ représentent des constantes vis-à-vis de la dérivée. Le second terme représente la dérivée de la sortie de la deuxième couche par rapport au paramètre $w_{2,(i,j)}$. Cette dernière est facilement calculable puisque seul le j^{eme} neurone de la couche 2 dépend de ce paramètre.

24 Rappelé en Annexe

La formule a été appliquée pour le cas particulier où $k = 2$. Pour $k = 1$ cela aurait donné :

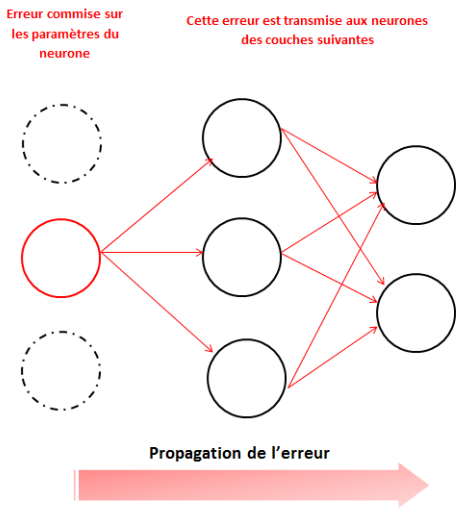
$$\frac{\partial F_W^{(1)}}{\partial w_{k=1,(m,n)}}(X_i) = \sum_{q=1}^{p_1=3} \frac{\partial_1 F_{W_3} \circ F_{W_2}}{\partial e_q}(F_{W_1}(X_i)) \times \frac{\partial_q F_{W_1}}{\partial w_{1,(m,n)}}(X_i).$$

En réitérant le théorème de dérivation des fonctions composées, on remarque de plus que :

$$\forall q \in [[1,3]], \frac{\partial_1 F_{W_3} \circ F_{W_2}}{\partial e_q}(z) = \sum_{l=1}^{p_2=4} \frac{\partial_1 F_{W_3}}{\partial e_l}(F_{W_2}(z)) \times \frac{\partial_l F_{W_2}}{\partial e_q}(z).$$

La décomposition de ces formules fait apparaître une forme de récurrence dans la façon d'écrire les dérivées partielles de chaque couche de neurones. A chaque fois, elles se décomposent en somme de produits de deux termes : une dérivée dépendant des paramètres $w_{k,(m,n)}$ et une dérivée dépendant des paramètres d'entrée de la couche. Cette décomposition introduit un moyen récursif de calculer le gradient (appelé *backpropagation*) sans devoir, à chaque ajout ou retrait d'une couche, calculer explicitement la fonction de transfert globale et ses multiples dérivées partielles. Afin d'introduire le principe de *backpropagation* illustrons par la Figure 21 le principe de propagation. Supposons qu'une erreur soit commise dans l'estimation d'un paramètre associé à un neurone. Ce neurone étant relié à la couche de sortie au travers de différentes couches, l'erreur est alors propagée dans chacun des neurones suivant le neurone erroné.

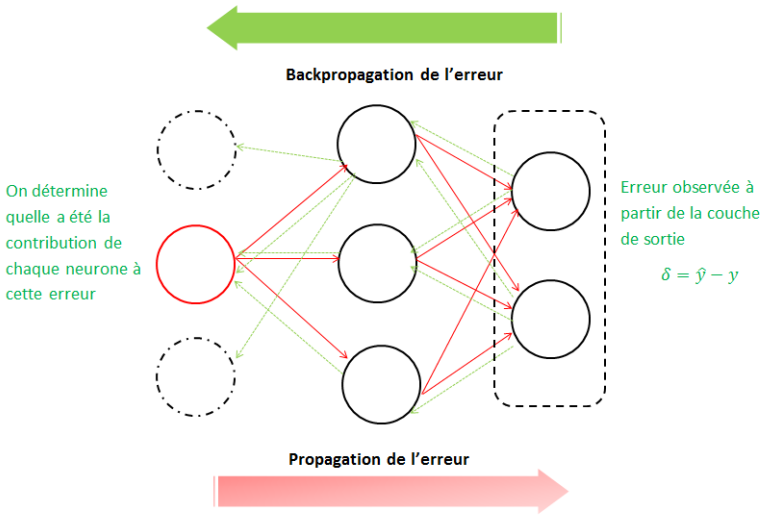
Figure 21.: Propagation de l'erreur d'estimation



Néanmoins, ce qui est facilement observable dans notre modèle neuronal c'est la sortie $\hat{y} = F_W(X)$ fournie par le réseau de neurone pour une entrée X donnée. Connaissant

l'erreur commise sur la prédiction, c'est-à-dire $\delta = \hat{y} - y$, nous aimererions corriger cette erreur en la rétroprogeant dans le réseau pour ainsi corriger les paramètres à l'origine de celle-ci. Cela revient à remonter à "contre courant" le schéma de la propagation précédent comme l'illustre la Figure 22. L'idée est donc d'utiliser l'erreur ou autrement dit, la sensibilité de la couche de sortie pour pouvoir estimé l'erreur de l'avant dernière couche et ainsi de suite. On préfère ainsi parler de rétropropagation ou *backpropagation* dans le cadre des réseaux de neurones.

Figure 22.: Propagation de l'erreur d'estimation



Pour faire le lien avec les calculs précédemment introduits. Une interprétation de la dérivée d'une fonction par rapport à ses coordonnées peut être énoncée comme telle : "Si je modifie légèrement l'entrée de ma fonction, comment va réagir la sortie. Ou autrement dit, quelle erreur cela va-t-il engendrer ?". Sur la base de cette idée, rappelons l'exemple précédent où l'on s'intéressait à l'erreur commise par un paramètre de la couche 2 soit :

$$\frac{\partial F_W^{(1)}}{\partial w_{k=2,(m,n)}}(X_i) = \sum_{q=1}^{p_2=4} \frac{\partial_1 F_{W_3}}{\partial e_q}(F_{W_2} \circ F_{W_1}(X_i)) \times \frac{\partial_q F_{W_2} \circ F_{W_1}}{\partial w_{2,(m,n)}}(X_i),$$

le terme $\frac{\partial_1 F_{W_3}}{\partial e_q}$ n'est autre que l'erreur commise par la couche 3. Si l'on récapitule, pour pouvoir développer un réseau de neurones modulaire, chaque couche doit être considérer comme un objet indépendant qui doit être en mesure de :

- Fournir un vecteur de sortie en fonction d'une entrée (éventuellement transmise par une autre couche);

- Calculer la dérivée partielle de sa fonction transfert en fonction des paramètres d'entrée et de la dérivée partielle de la couche suivante;
- Calculer la dérivée partielle de sa fonction transfert par rapport aux paramètres $w_{k,(m,n)}$ qu'elle contient;
- De mettre à jour les paramètres en fonction de l'erreur globale observée.

L'intérêt d'utiliser un langage orienté objet comme Python prend alors tout son sens dans ce type d'implémentation. Il permet en effet d'établir facilement des architectures de réseaux de neurones différentes tout en minimisant les modifications de code. On présente ci-dessous l'architecture du module codé en Python[25]. Il est en effet important de souligner que l'aspect technologique est indissociable de l'apprentissage statistique. Le temps économisé dans l'implémentation des algorithmes permet alors d'affiner la robustesse du modèle et de pouvoir en tester plusieurs de façon simple. D'autre part la maîtrise de l'outil permet de comprendre au mieux le fonctionnement d'un réseau de neurones et de ne pas être sujet à la problématique de "boite noire" de nombreux algorithmes.

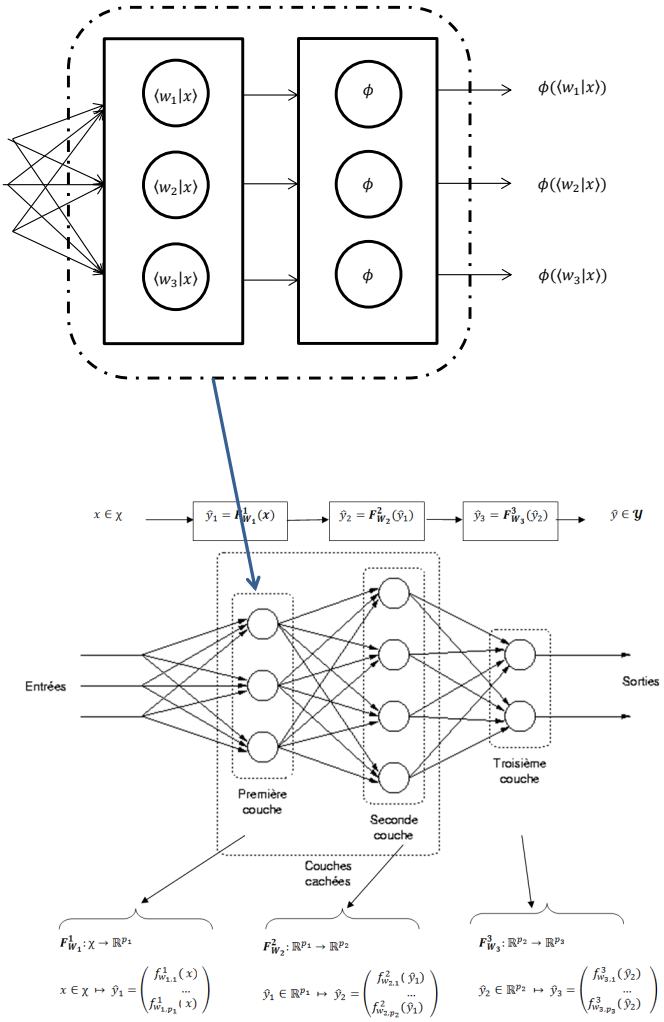
```
class Module:
    #Permet le calcul de la sortie du module
    def forward(self,input_x):
        pass

    #Permet le calcul du gradient des cellules d'entree
    def backward_delta(self,input_x,delta_module_suivant):
        pass

    #Permet d'initialiser le gradient du module
    def init_gradient(self):
        pass

    #Permet la mise a jour des parametres du module
    #avec la valeur courante du gradient
    def update_parameters(self,gradient_step):
        pass

    #Permet de mettre a jour la valeur courante du gradient par addition
    def backward_update_gradient(self,input,delta_module_suivant):
```

25 Même si le développement a entièrement été effectué par nos soins, l'architecture initiale s'inspire du cours d'Apprentissage Statistique donné par Ludovic Denoyer et Patrick Gallinari (UPMC) que nous avons suivi dans le cadre du Master Données Apprentissage et Connaissance

```
        pass

    #Permet de faire les deux backwards simultanement
    def backward(self,input,delta_module_suivant):
        self.backward_update_gradient(input,delta_module_suivant)
        return self.backward_delte(input,delta_module_suivant)

    #Retourne les parametres du module
    def get_parameters(self):
        pass

    #Initialise aleatoirement les parametres du module
    def randomize_parameters(self, variance):
        pass
```

Ainsi nous avons développé plusieurs modules dont le nombre de neurones est facilement modifiable. Nous avons plus particulièrement codé un module de neurones linéaires (i.e. Φ, fonction d'activation, est égale à la fonction identité), un module de fonctions d'activation tangente hyperbolique et un module sigmoïde (comme défini précédemment). L'avantage d'une vision modulaire est que les fonctions d'activation peuvent être vues comme une couche indépendante de neurones sans paramètre à calibrer. Il est ainsi facile d'établir différentes architectures qui s'adaptent aux algorithmes d'optimisation.

Figure 23.: Schéma de l'implémentation modulaire d'une couche



2.3 APPLICATION DES RÉSEAUX DE NEURONES À DES FINS DE RÉGRESSIONS

Les réseaux de neurones étant peu utilisés à des fins de régressions (cf paragraphe 2.2.1), nous avons souhaité tester leurs performances dans la détermination du capital économique requis à un an. La réglementation Solvabilité II impose aux compag-

nies d'assurance de posséder suffisamment de fonds propres pour faire face à une ruine économique dans un an avec une probabilité de 99,5%. Dans ce contexte, nous avons pu voir en première partie de ce mémoire que la méthode de référence était celle des Simulations dans les Simulations. Cependant, dues aux nombreuses interactions entre les actifs et les passifs, cette méthode est trop coûteuse en calcul ce qui n'autorise que la simulation de quelques centaines de fonds propres à un an.

Une première alternative consiste donc à partir de l'observation de ces quelques points d'estimer la fonction de la valeur économique du portefeuille à un an (*curve-fitting*). Une deuxième méthode introduite en première partie consiste à utiliser la méthode des *Least Squares Monte Carlo* qui permet de construire une fonction qui à partir des facteurs de risque de première période estime la valeur moyenne actualisée des fonds propres à un an.

$$\mathbb{E}^{\mathbb{Q}}(VAN \mid \epsilon) = VEP_1 = f(\epsilon) \tag{4}$$

Nous avons alors étudié le comportement des réseaux de neurones dans ces deux cas. L'objectif est d'étudier dans quelle mesure les modèles neuronaux peuvent fournir une estimation de la valeur économique des fonds propres à un an. Pour ce faire, avant de confronter directement le modèle d'apprentissage statistique aux méthodes d'estimation les plus pratiquées actuellement en assurance (à savoir les méthodes polynomiales) sur des données réelles, nous nous sommes intéressés à un cas d'étude.

2.3.1 *Présentation des données*

Nous travaillons dans un premier temps avec des données simulées avec le souci de conserver l'homogénéité des objets manipulés avec ceux utilisés par les opérationnels (valeurs économiques de portefeuille ou flux monétaires). Pour revenir sur les enjeux de la *datascience* exposés en partie 2, on s'affranchit dans cette partie de la phase de Data Management (Figure 5 page 34) qui est une part importante dans le processus de traitement des données. Nous nous focalisons dans ce qui va suivre sur l'intégration des méthodes d'apprentissage statistique dans la gestion des risques et plus spécifiquement dans le pilotage des risques financiers.

On considère $N = 10$ indices d'action que nous noterons $(S_1, ..., S_{10})$. On note r un taux d'intérêt sans risque que nous supposons être constant au court du temps. Sous probabilité historique $\mathbb{P}$ nous diffusons en première période nos 10 indices $(S_1, ..., S_{10})$ selon un modèle Black-Scholes dont on rappelle la formule (l'exposant (1) marque la première période $t = 1$) :

$$S_1^{(1),p} = e^{[(r-\frac{\sigma_1^2}{2})+\sigma_1 W_{1,p}]}$$

où W_1 représente une variable aléatoire suivant une loi normale $\mathcal{N}(0,1)$.

De ces dix indices actions nous constituons un portefeuille de 10 *calls* dont les variances différent et où les strikes $K_1, ..., K_{10}$ sont choisis entre 75% et 125%. Afin de simplifier notre étude, nous avons supposé les maturités T des différentes options égales à 2 ans. Le tableau ci-dessous récapitule les paramètres financiers utilisés dans notre cas d'étude.

Table 6.: Paramètres financiers des *Calls*

Sous-jacent	Valeur du taux sans risque r	σ	Strike K
S_1	0.05	0.305	0.932
S_2		0.0396	1.0143
S_3		0.121	1.005
S_4		0.193	0.791
S_5		0.0544	0.981
S_6		0.199	0.856
S_7		0.202	0.946
S_8		0.186	1.049
S_9		0.253	0.764
S_{10}		0.0607	1.058

Enfin, afin d'apporter plus de convexité à notre fonction cible et également avoir moins de monotonie nous pondérerons les différents *Calls*. On introduit ainsi les poids $(w_1, .., w_{10}) \in \mathbb{R}^{10}$ choisis aléatoirement non nuls de sorte à avoir un portefeuille dont la formule de valorisation en date t est la suivante :

$$f(S_1^{(t)}, ..., S_{10}^{(t)}) = \sum_{k=1}^{10} w_k * Call_k(t, r, S_k^{(t)}, K_k, T = 2)$$

où pour tout entier $k \in [\![1, 10]\!]$, $Call_k(t, r, S_k^{(t)}, K_k, T = 2)$, représente le prix Black-Schole d'un *Call* à la date t dont le sous-jacent est S_k, la maturité $T = 2$, le strike K_k.

Table 7.: Valeurs des poids w_k

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
-1.41	-0.52	-1.54	-1.58	2.08	-1.94	3.04	4.45	0.86	-1.98

On rappelle la formule de prix Black-Scholes utilisée ici pour tout $k \in [\![1, 10]\!]$:

$$Call_k(t, r, S_k^{(t)}, K_k, T = 2) = S_k^{(t)} \phi(d1) - K_k e^{-r} \phi(d2)$$

avec

$$d1 = \frac{r + \frac{\sigma_k^2}{2}(T-t) + ln(\frac{S_k^{(t)}}{K_k})}{\sigma_k(T-t)}, \qquad d_2 = \frac{r - \frac{\sigma_k^2}{2}(T-t) + ln(\frac{S_k^{(t)}}{K_k})}{\sigma_k\sqrt{T-t}}.$$

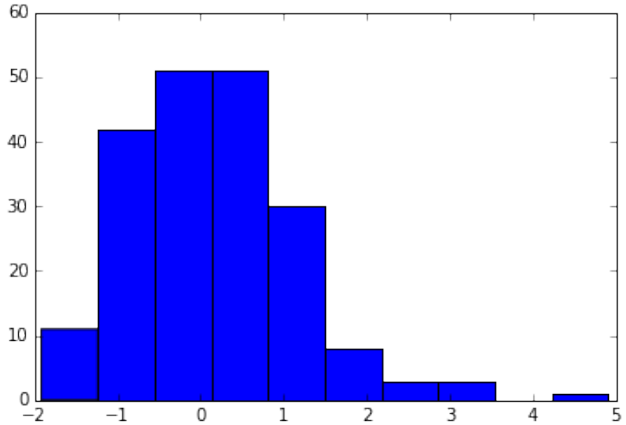
2.3.2 *Les réseaux de neurones dans un cadre de curve-fitting*

Dans cette étude, la problématique est d'estimer f fonction du vecteur de facteurs de risque de première période $\epsilon \in \mathbb{R}^{10}$ modélisant la valeur économique des passifs à un an en n'observant que la valeur de cette fonction qu'en quelques points. Nous nous plaçons dans le cas où l'assureur a pu simuler 200 scénarios de valeurs économiques par la méthode des Simulations dans les Simulations. Ces valeurs étant homogènes à un prix, on observe alors $n = 200$ valeurs du portefeuille $f(S_1^{(t)}, ..., S_{10}^{(t)})$ en date $t = 1$.

Dans ce problème notre base d'apprentissage comporte alors $n = 200$ couples (y_i, X_i). Pour chacun des scénarios i, y_i représente la valeur du portefeuille $f(S_1^{(1)}, ..., S_{10}^{(1)})$ associée à la valeur $X_i = (\epsilon_{1,i}, ..., \epsilon_{10,i}) = (\sigma_1 W_{i,1}, ..., \sigma_{10} W_{i,10})$ où pour tout scénario i les $W_{i,k}$ représentent des variables aléatoires indépendantes suivant une loi normale $\mathcal{N}(0, 1)$.

Afin de posséder suffisamment de recul sur les résultats de nos modèles, nous exposons une première étude descriptive des données simulées. L'histogramme de la Figure 24 présente la répartition des y_i. On constate que cette distribution ne possède pas de symétrie particulière et présente une queue de distribution relativement épaisse. Ce point peut être déterminant dans le calibrage du réseau de neurones. Une trop forte variance sur la variable à expliquer peut en effet rendre le calibrage difficile.

Figure 24.: Histogramme de répartition des 200 valeurs y_i



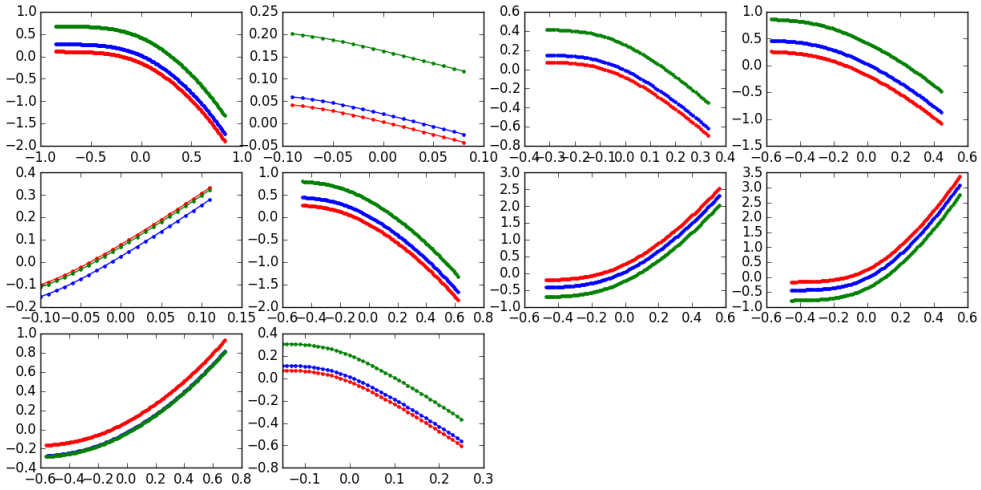
Le tableau 8 présente la répartition des valeurs de marché prises par le portefeuille *calls* constituant notre portefeuille.

Table 8.: Description de la répartition des valeurs prises par les *calls*

Call	1	2	3	4	5	6	7	8	9	10
moyenne	0.258	0.087	0.115	0.323	0.120	0.259	0.209	0.145	0.385	0.062
écart-type	0.255	0.038	0.098	0.203	0.048	0.182	0.191	0.139	0.216	0.050
min	0.001	0.013	0.001	0.006	0.032	0.009	0.003	0.001	0.017	0.001
quartile 1	0.067	0.060	0.039	0.158	0.086	0.141	0.074	0.041	0.227	0.029
médiane	0.183	0.089	0.085	0.287	0.113	0.231	0.148	0.103	0.359	0.049
quartile 3	0.375	0.112	0.172	0.454	0.155	0.347	0.282	0.209	0.516	0.082
max	1.441	0.183	0.506	0.860	0.251	1.111	0.917	0.814	1.295	0.349

Afin d'analyser au mieux la qualité d'approximation de la fonction f par nos modèles nous avons étudié les sensibilités marginales de notre fonction cible par rapport aux différents facteurs de risque $\epsilon_1, ..., \epsilon_{10}$. Cette étape consiste à analyser le comportement de la fonction f lorsque toutes choses égales par ailleurs, une seule des composantes du vecteur X_i varie. Nous avons fixé dans chacun des cas les autres paramètres aux trois premiers quartiles (rouge : premier quartile, bleu : médiane et vert : troisième quartile) et tracé le comportement de f. On obtient alors la Figure 25 où chacune des courbes représente l'évolution marginale de la fonction f en fonction de ϵ_k pour $k \in [\![1, 10]\!]$ (ces courbes sont exposées dans l'ordre de gauche à droite puis de haut en bas).

Figure 25.: Etude des sensibilités des valeurs de f en fonction des facteurs de risque en abscisse. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



Enfin, nous confronterons le modèle neuronal à un modèle couramment utilisé en assurance. Nous analyserons les effets marginaux des deux modèles et évaluerons leurs performances prédictives sur un échantillon de test de 50 nouveaux points.

2.3.2.1 *Calibrage d'un modèle polynomial*

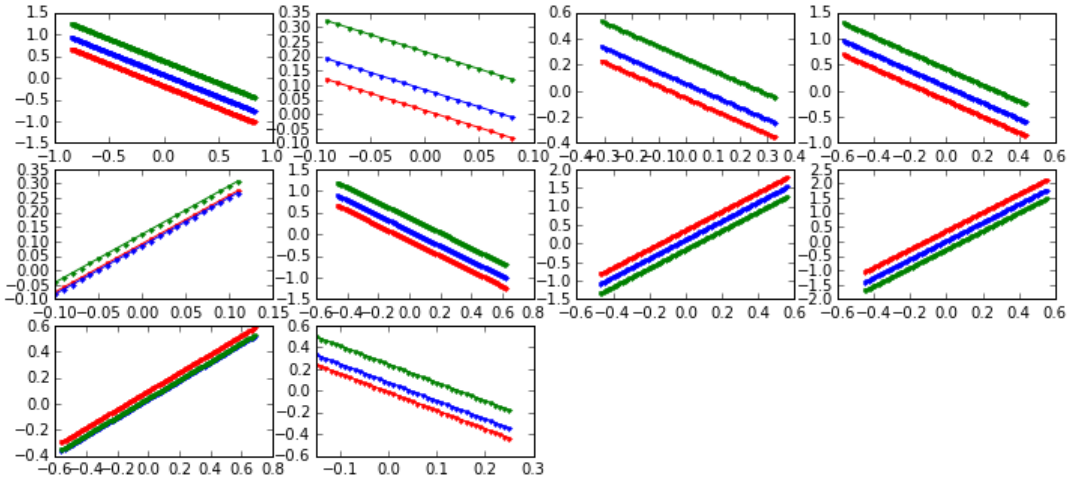
En tant que première modélisation, nous avons adopté une approche polynomiale. Cette méthode est actuellement une des plus utilisées en assurance vie afin d'approcher une fonction inconnue. Elle part du théorème d'analyse de Stone-Weierstrass qui énonce que l'ensemble des fonctions polynomiales est dense dans l'ensemble des fonctions continues. En d'autres termes, toute fonction continue peut être approchée par un polynôme dont le degré est plus ou moins élevé. Cette méthode est relativement simple et bien maîtrisée par les actuaires dans la mesure où le calibrage se ramène à régression linéaire standard.

L'idée est donc de confronter la capacité d'approximation fonctionnelle des polynômes à celle des réseaux de neurones. Nous considérons dans un premier temps le modèle le plus simple :

$$\hat{poly}_1(\epsilon_1, ..., \epsilon_{10}) = \alpha_0 + \sum_{k=1}^{10} \alpha_k \epsilon_k.$$

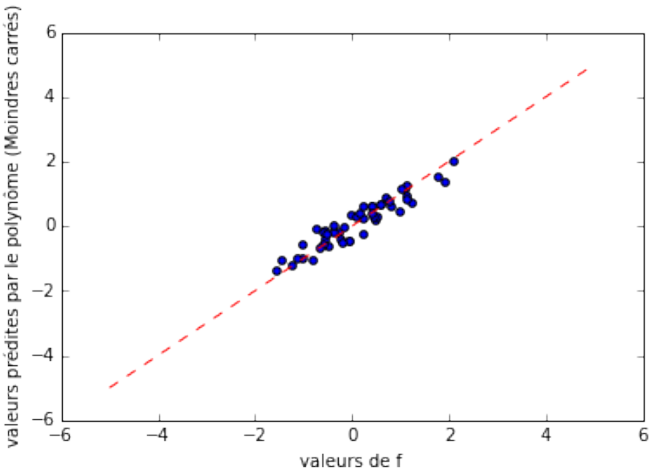
Ce premier modèle n'est autre qu'une régression linéaire ordinaire réalisée à partir des $n = 200$ couples de points (y_i, X_i) où l'on rappelle que $X_i = (\epsilon_{1,i}, ..., \epsilon_{10,i})$. Ce modèle est donc linéaire aussi bien en les paramètres α_k qu'en les facteurs de risque ϵ_k. En termes d'effets marginaux représentés en Figure 26 même si la courbure est éloignée des graphiques tracés en Figure 25, la monotonie reste conservée.

Figure 26.: Etude des sensibilités de $\hat{pol}y_1$ (en ordonnée) en fonction des facteurs de risque en abscisse. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



En ce qui concerne la capacité prédictive du modèle, les résultats sont discutables pour ce polynôme. La Figure 27 représente la valeur fournie par le *proxy* à partir des données de la base de test au regard de la vraie valeur de la fonction f. On constate que la variance de l'erreur de prédiction est assez élevée. Afin de comparer la précision de ce modèle avec les autres par la suite on retiendra également que l'erreur quadratique sur cette échantillon de test est de **0,0781**.

Figure 27.: Tracé des valeurs prédites sur la base de test en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)

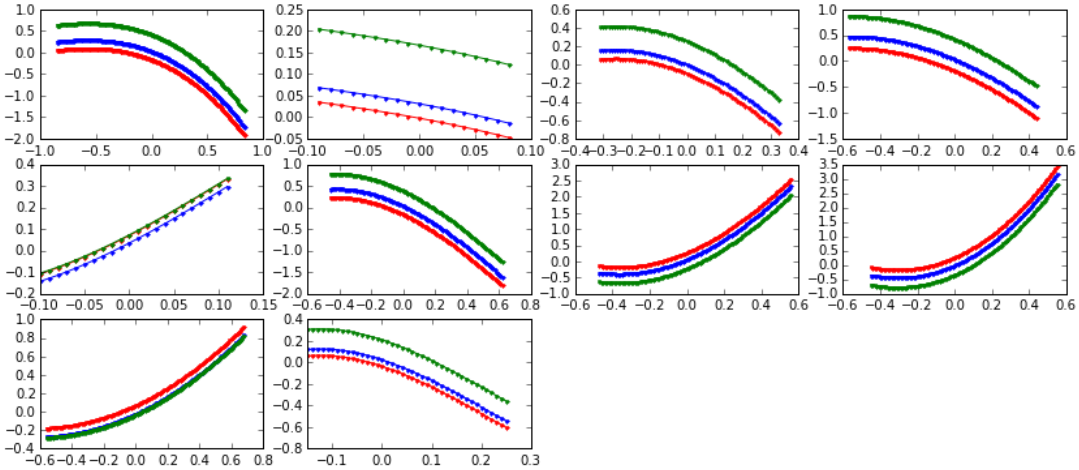


Cependant, il est commun pour les utilisateurs de *proxies* polynomiaux de considérer également les variables explicatives élevées à la puissance 2 et 3 dans la construction du modèle. Pour cette raison nous avons à des fins de comparaisons également étudié le modèle suivant :

$$\hat{poly}_2(\epsilon_1, ..., \epsilon_{10}) = \alpha_0 + \sum_{k=1}^{10} (\alpha_k \epsilon_k + \beta_k \epsilon_k^2 + \omega_k \epsilon_k^3).$$

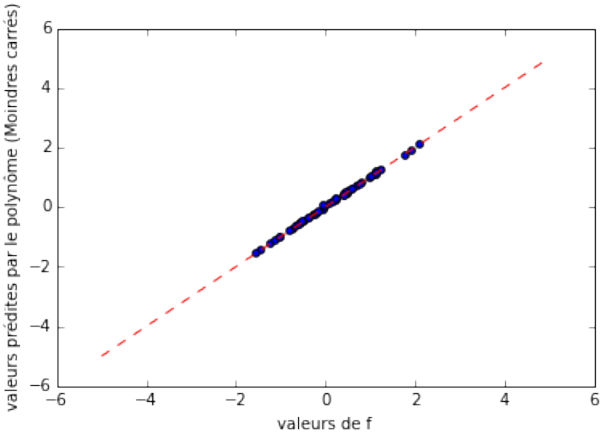
Les différents facteurs ϵ sont élevés à différentes puissances (1, 2 et 3) et les paramètres $\alpha_k, \beta_k, \omega_k$ pour $k \in [\![1, 10]\!]$ sont déterminés également par moindres carrés. Dans ce modèle, les termes croisés ne sont pas pris en compte. Cette omission se justifie par la constitution du portefeuille de *calls*. Ces derniers dépendent en effet tous de sous-jacents supposés indépendants et la valeur du portefeuille que l'on souhaite répliquer est prise comme la somme pondérée des valeurs des *calls*. Comme le montre la Figure 28, les résultats obtenus par cette approche sont alors beaucoup plus satisfaisants.

Figure 28.: Étude des sensibilités de $\hat{polY}_2$ (en ordonnée) en fonction des valeurs prises par les différents facteurs de risque. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



A vue d'œil, le comportement marginal de notre polynôme semble être très proche du celui observé sur la fonction f. Lorsque l'on étudie les prédictions faites par ce dernier modèle sur la base de test, on obtient alors une erreur quadratique très faible de **0,000608**. La Figure 29 confirme le bon pouvoir prédictif de cette fonction sur l'échantillon de test.

Figure 29.: Tracé des valeurs prédites par $\hat{polY}_2$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)



Étant donnés les résultats obtenus sur l'échantillon de test, il semblerait qu'un polynôme avec les monômes des degrés 2 et 3 semblent correctement approcher f. L'introduction de variables explicatives supplémentaires nous aurait laissé penser que le polynôme manifesterait un cas de sur-apprentissage sur la base de test. Le modèle comporte en effet 31 variables explicatives pour 200 observations dans la base d'apprentissage. A ce stade, on peut donc émettre une première critique quant à la fonction f à savoir que sa convexité et sa non linéarité n'est pas aussi complexe que dans la réalité opérationnel. Ceci expliquerait les bons résultats obtenus par $\hat{pol}y_2$ alors que nous avons expérimenté en entreprise qu'il est difficile d'utiliser ce type de modèle sur la base de données réelles.

2.3.2.2 *Calibrage d'un modèle neuronal*

Dans ce paragraphe nous avons testé plusieurs structures de réseaux de neurones. Comme introduit précédemment, il est en effet laissé à l'utilisateur le choix de la structure ainsi que la détermination des méta-paramètres (nombre d'itérations de calibrage, méthode de gradient, pas de gradient). L'objectif est de comparer les résultats des réseaux de neurones avec les modèles polynomiaux précédemment introduits. Ainsi on testera des modèles neuronaux où l'entrée est le vecteur de facteurs de risque (de dimension 11 : 10 facteurs de risque et la constante designée par ϵ_{11}) et d'autres où l'entrée tient compte des facteurs élevés à la puissance 2 et 3 comme dans le second polynôme. Guidé par quelques éléments théoriques présenté en amont (Théorème de Cybenko [Cybenko 1989]), nous avons donc testé différentes structures de la plus simple à la plus complexe.

Cas où l'entrée est le vecteur de facteur de risque

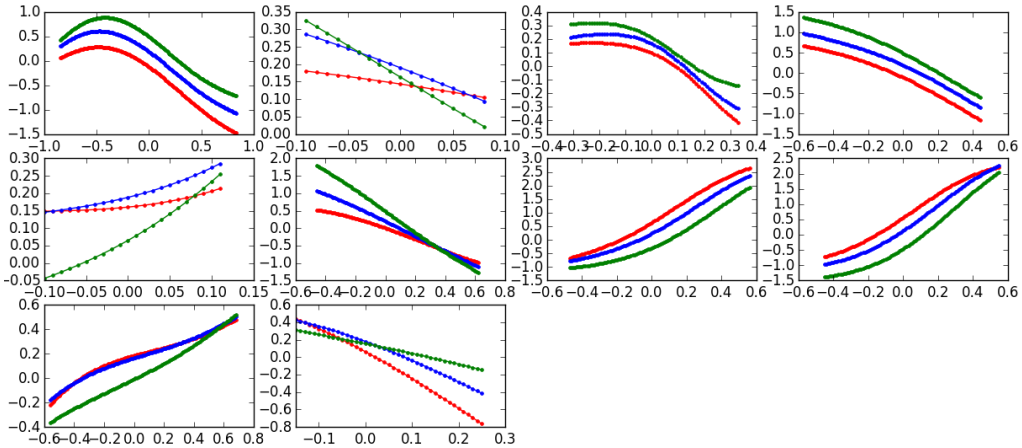
Réseau de neurones à 1 couche, 20 neurones Tangente Hyperbolique
En première approche, nous avons évalué les performances d'un réseau de neurones à une couche. Cette couche est constituée de 20 neurones dont les fonctions d'activation sont des fonctions Tangente Hyperbolique (définies dans le Tableau 3 page 48). Formellement ce modèle peut s'écrire :

$$\forall \epsilon \in \mathbb{R}^{11},\ nnet_1(\epsilon) = \sum_{i=1}^{20} \alpha_i Tanh(w_i^T \epsilon)$$

où pour tout $i \in [\![1, 20]\!]$, les réels α_i ainsi que les vecteurs réels $w_i \in \mathbb{R}^{11}$ sont des paramètres à calibrer.

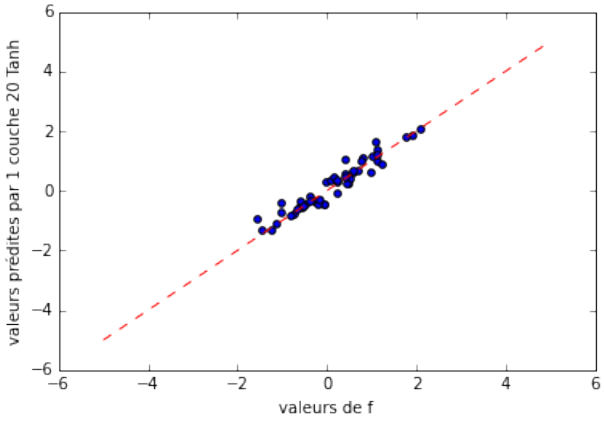
Ce réseau a été calibré par méthode de descente de gradient stochastique avec contrôle du gradient selon la méthode Adagrad présentée précédemment. Le nombre d'itérations nécessaires au calibrage a quant à lui été déterminé par cross-validation. Dans le cas de cette structure d'une couche à 20 neurones, ce nombre est de 100 itérations. La structure des sensibilités de ce modèle est donnée en Figure 30.

Figure 30.: Étude des sensibilités de $nnet_1$ en fonction des facteurs de risque (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



Si la monotonie des sensibilités est similaire à celle de la fonction f que l'on souhaite approcher, on constate que certaines des sensibilités présentent des changements de convexité alors que celles de la fonction f n'en possèdent pas. Cependant, l'erreur quadratique sur la base de test de ce premier modèle neuronale est de **0,0587**. La Figure 31 présente les résultats prédits sur la base de test. Au regard des performances obtenues par le modèle polynomial $\hat{polY}_1$ qui dépend exactement des mêmes paramètres d'entrée, le réseau $nnet_1$ est meilleur. Cependant, ce résultat est moins appréciable si on le compare aux résultats du modèle polynomial $\hat{polY}_2$ qui prend en compte les facteurs de risque élevés à la puissance 2 et 3. Afin d'avoir un meilleur aperçu des résultats obtenus par les modèles, un tableau synthétisant les performances sur la base de test est présenté ci-après en page 93.

Figure 31.: Tracé des valeurs prédites par $nnet_1$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)



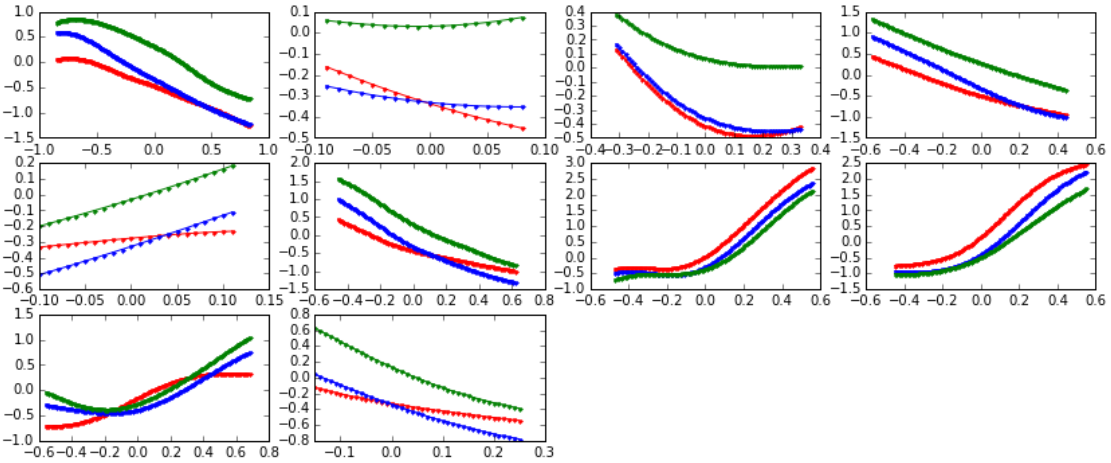
Réseau de neurones à 1 couche, 50 neurones Tangente Hyperbolique
Dans une seconde configuration, nous avons testé un réseau de neurones à une couche contenant 50 neurones avec des fonctions d'activation Tangente Hyperbolique. Le modèle s'écrit alors :

$$\forall \epsilon \in \mathbb{R}^{11},\ nnet_2(\epsilon) = \sum_{i=1}^{50} \alpha_i Tanh(w_i^T \epsilon)$$

où pour tout $i \in [\![1, 50]\!]$, les réels α_i ainsi que les vecteurs réels $w_i \in \mathbb{R}^{11}$ sont des paramètres à calibrer.

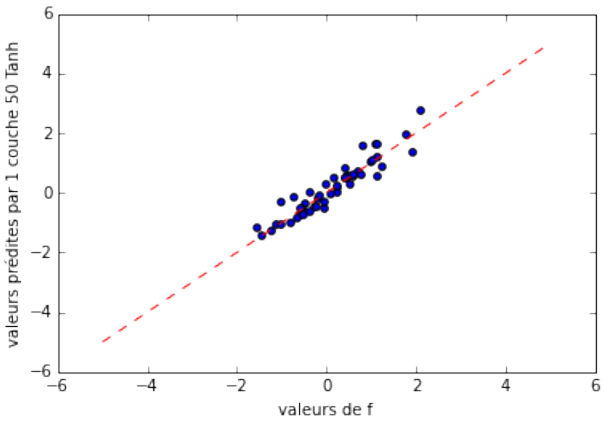
La méthode de calibrage la plus performante (c.a.d la plus faible erreur sur la base de test et un temps de calibrage faible) reste la méthode de gradient stochastique avec le contrôle de gradient selon la méthode Adagrad. Le nombre d'itérations est quant à lui de 350. On obtient les sensibilités présentées par la Figure 32.

Figure 32.: Etude des sensibilités de $nnet_2$ en fonction des facteurs de risque. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



Les résultats obtenus par ce modèle restent assez mitigés. L'augmentation du nombre de neurones semble engendrer un phénomène de sur-apprentissage. Les sensibilités ne conservent pas toutes la même monotonie que celles observées sur la fonction f à approcher. Les résultats de prédiction sur la base de test sont présentés en Figure 33. D'autre part, l'erreur quadratique sur la base de test est de **0.0926** ce qui représente la moins bonne des performances.

Figure 33.: Tracé des valeurs prédites par $nnet_2$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)



Réseau de neurones à 2 couches

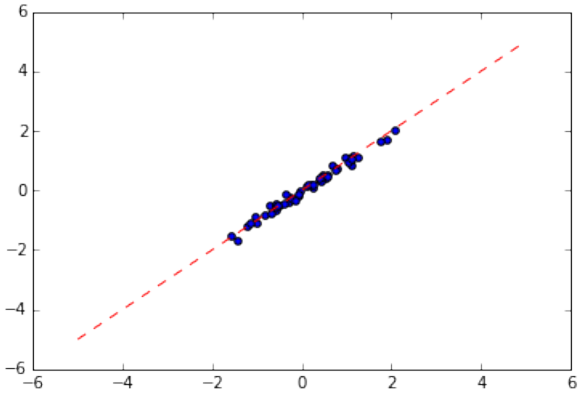
Nous avons enfin souhaité tester une architecture de réseau plus complexe : un réseau de neurones à 2 couches cachées. La première couche est constituée de 20 neurones de fonction d'activation Tangente Hyperbolique. La deuxième contient également 20 neurones mais cette fois avec une fonction d'activation Sigmoïde. La procédure de calibrage reste identique hormis le nombre d'itérations qui, suite à une procédure de *cross-validation*, a été fixé à 500. Empiriquement on constate que la complexité (nombre de couches et de neurones par couche) du réseau tend à faire augmenter le nombre d'itérations nécessaires au calibrage. Le modèle ainsi testé peut se formaliser de la manière suivante :

$$\forall \epsilon = (\epsilon_1, ..., \epsilon_{11}) \in \mathbb{R}^{11},\ nnet_3(\epsilon) = \sum_{j=1}^{20}[\alpha_j Sigmoide(\sum_{p=1}^{20} \beta_{j,p} Tanh(\sum_{l=1}^{11} w_{p,l}\epsilon_l))]$$

où pour tout $j, p \in [\![1, 20]\!]$, $l \in [\![1, 11]\!]$ les réels $\alpha_j, \beta_{j,p}, w_{p,l}$ sont des paramètres à calibrer.

Les résultats obtenus par ce modèle semblent satisfaisants au regard de ceux obtenus par le polynôme $\hat{pol}y_1$ et des réseaux de neurones précédents $nnet_1$ et $nnet_2$. L'erreur quadratique sur la base de test est en effet de **0,0153** ce qui parmi tous les modèles cités précédemment est la meilleure performance obtenue (on récapitule les différents résultats dans le tableau 9 ci-après). La Figure 34 présente la courbe des prédictions obtenues.

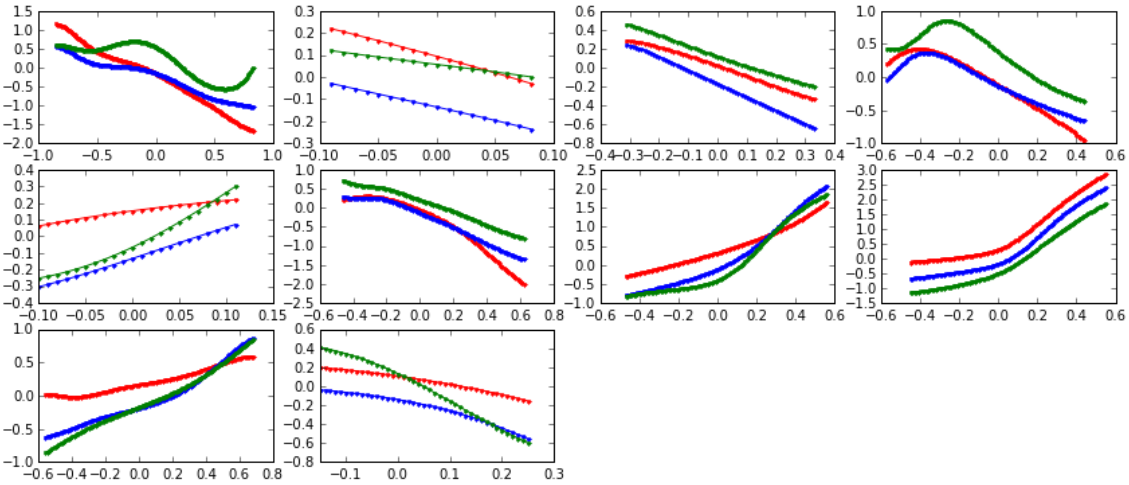
Figure 34.: Tracé des valeurs prédites par $nnet_3$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)



Malgré les bonnes performances prédictives obtenues par $nnet_3$ (relativement à $\hat{pol}y_1$, $nnet_1$ et $nnet_2$), l'analyse des effets marginaux tend à nuancer notre propos par les inversions de monotonies constatées sur certaines sensibilités comme l'illustre la Figure

35. Cette inversion de monotonie peut s'avérer en effet dangereuse dans la mesure où l'utilisation du réseau à des fins de prédiction pourrait amener à prédire une évolution de valeur de fonds propres allant dans le sens opposé à la variation réelle.

Figure 35.: Etude des sensibilités de $nnet_3$ en fonction des facteurs de risque. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



A ce stade, si l'on compare la méthode polynomiale et neuronale, à partir des mêmes vecteurs d'entrée (les facteurs de risque non modifiés), le modèle neuronal présente de meilleures performances (selon la mesure de l'erreur quadratique) sur la base de test. Cependant, le sur-paramétrage du réseau de neurones semble entraîner certaines distorsions des sensibilités marginales. On constate alors un phénomène de sur-apprentissage. Ce dernier point remet alors en cause les performances du réseau de neurones. Nous revenons sur ce point dans la partie suivante.

Cas où l'entrée sont les facteurs de risque élevés au degré 1, 2 et 3

Après avoir comparé les réseaux de neurones au modèle $\hat{pol}y_1$, nous avons réitéré la démarche pour $\hat{pol}y_2$. En considérant les facteurs de risque $(\epsilon_1, ..., \epsilon_{10}, \epsilon_1^2, ..., \epsilon_{10}^2, \epsilon_1^3, ..., \epsilon_{10}^3)$ auquel on ajoutera le facteur constant $\epsilon_{11} = 1$, nous testons les performances prédictives des structures neuronales testées en amont. Suite aux résultats précedemment exposés, on notera que le modèle polynomial $\hat{pol}y_2$ présente d'ores et déjà de très bonnes performances.

Réseau de neurones à 1 couche, 20 neurones Tangente Hyperbolique
Le premier réseau de neurones possède une structure identique à $nnet_1$ présenté précédemment. La seule différence réside dans la dimension du vecteur d'entrée qui au lieu d'appartenir à $\mathbb{R}^{11}$ (dix facteurs de risque et une constante) appartient cette fois à $\mathbb{R}^{31}$ (dix facteurs de risque élevé à la puissance 1, 2 et 3 et une constante). Etant donnée cette similitude dans les structures, nous conserverons alors des notations proches pour faire référence aux structures de réseau de neurones identiques. Ainsi nous noterons le réseau de neurones à une couche de 20 neurones dont l'entrée est un vecteur de $\mathbb{R}^{31}$ $\tilde{nnet_1}$ dont nous rappelons la définition :

$$\forall \epsilon \in \mathbb{R}^{31},\ \tilde{nnet_1}(\epsilon) = \sum_{i=1}^{20} \alpha_i Tanh(w_i^T \epsilon)$$

où pour tout $i \in [\![1, 20]\!]$, les réels α_i ainsi que les vecteurs réels $w_i \in \mathbb{R}^{31}$ sont des paramètres à calibrer.

Compte-tenu du nombre de paramètres à calibrer plus grand, ce réseau a nécessité 150 itérations dans la procédure de calibrage (contre 100 pour $nnet_1$). Les performances obtenues pour cette structure sont assez mauvaises. L'erreur quadratique (illustrée par la Figure 36) sur la base de test est de **0.0903**. D'autre part, l'étude des sensibilités qu'illustre la Figure 37 confirme la supériorité des performances du modèle $\hat{poly_2}$. Ces dernières présentent des courbures et monotonies différentes de celles de la fonction f à approcher.

Figure 36.: Tracé des valeurs prédites par $\tilde{nnet_1}$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)

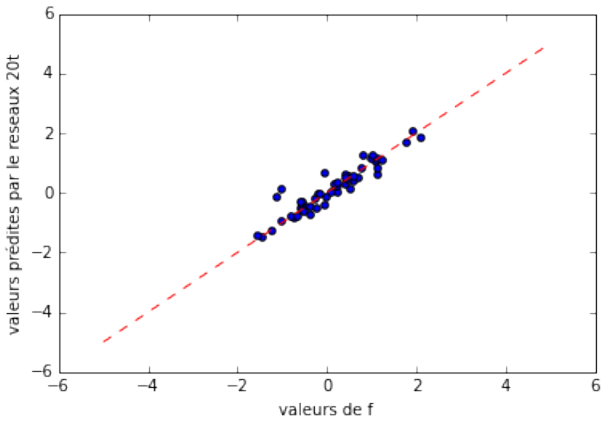
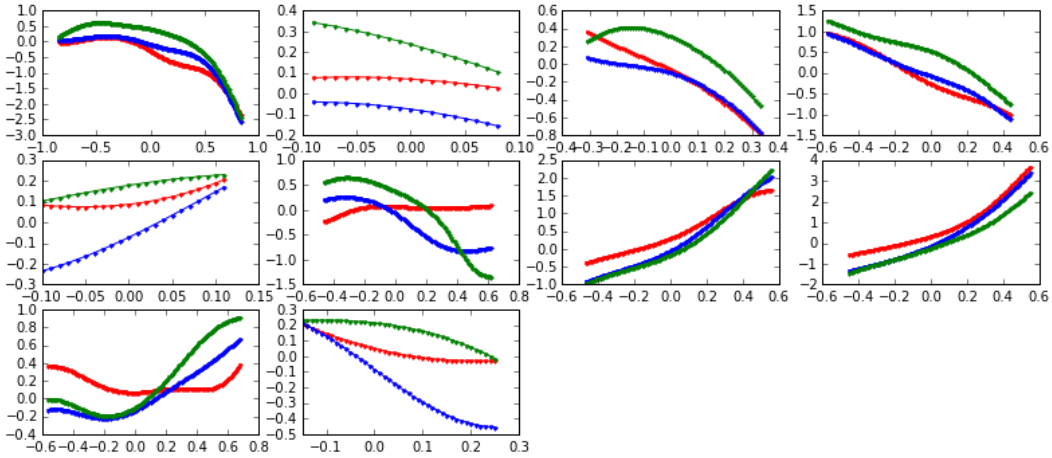


Figure 37.: Etude des sensibilités de $\tilde{nnet}_1$ en fonction des facteurs de risque. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



Réseau de neurones à 1 couche, 50 neurones Tangente Hyperbolique
Augmenter le nombre de neurones dans la couche cachée permet de réduire l'erreur quadratique (illustrée par la Figure 38) constatée sur la base de test à **0,0251** . Seulement cette erreur reste nettement supérieure à celle obtenue par le modèle $\hat{poly}_2$ qui on le rappelle était de **0,000608**. Enfin l'analyse des sensibilités (Figure 39) du modèle à 50 neurones $\tilde{nnet}_2$ ne présente pas non plus de résultats au moins aussi satisfaisants que $\hat{poly}_2$ exposé en page 82.

Figure 38.: Tracé des valeurs prédites par $\tilde{nnet}_2$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)

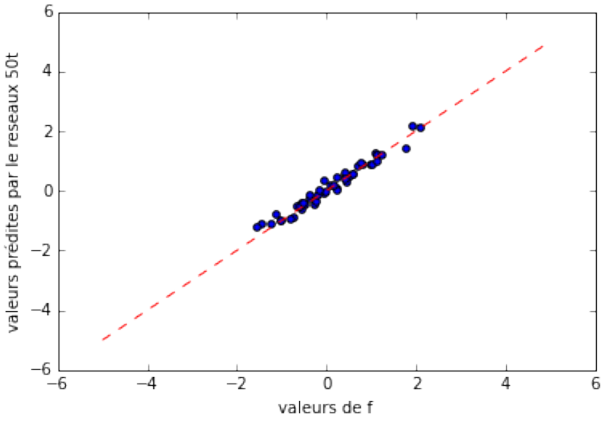
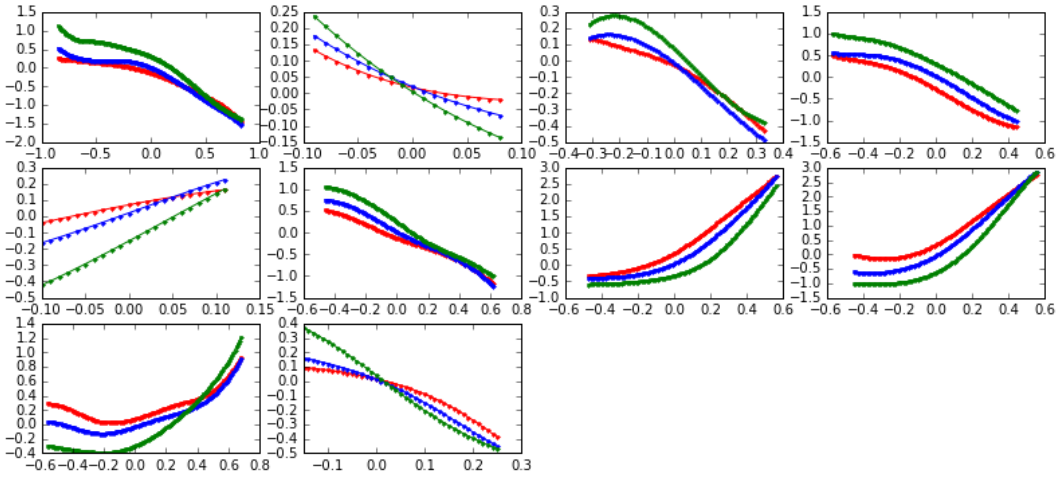


Figure 39.: Etude des sensibilités de $\tilde{nnet_2}$ en fonction des facteurs de risque. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



Réseau de neurones à 2 couches

Enfin, le réseau à deux couches cachées que nous notons $\tilde{nnet_3}$ diminue légèrement l'erreur quadratique (illustrée par la Figure 40) observée sur la base de test. Nous arrivons à **0,0220** soit 12% de moins que l'erreur du réseau de neurones monocouche à 50 neurones que nous avions noté $\tilde{nnet_2}$. Même si ce modèle présente les meilleures performances parmi les structures de réseaux de neurones testées, cela est moins satisfaisant que les résultats de $\hat{poly_2}$. On remarque d'autre part, que les sensibilités de $\tilde{nnet_3}$ présentées en Figure 41 comportent de nombreux changements de monotonie par rapport aux effets marginaux réels constatés dans l'étude de la fonction f que l'on souhaite approcher. Ce dernier point soulève donc de nombreuses réserves quant à l'utilisation de ce réseau à des fins de *curve-fitting* pour la détermination de fonds propres à un an.

Figure 40.: Tracé des valeurs prédites par $n\tilde{n}et_3$ en fonction des vraies valeurs (la droite en pointillés représente la prédiction parfaite)

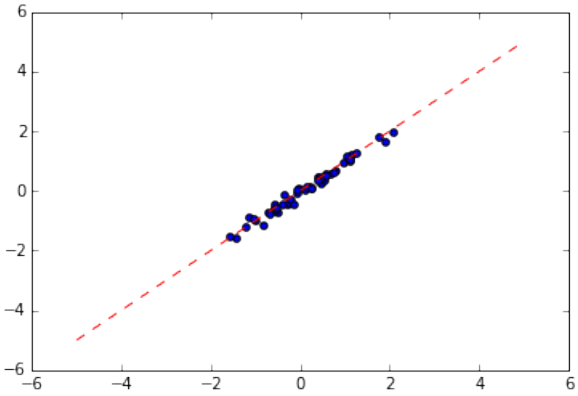
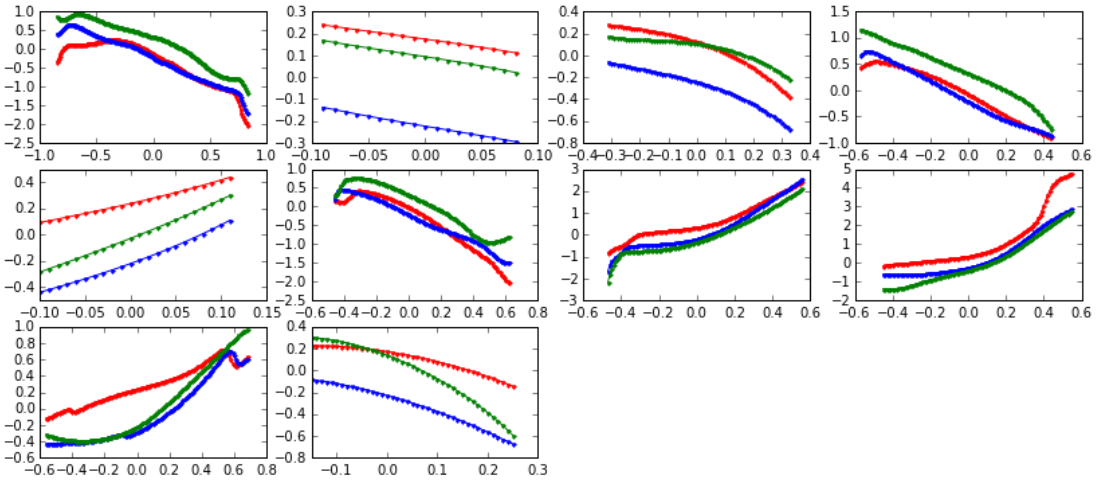


Figure 41.: Etude des sensibilités de $n\tilde{n}et_3$ en fonction des facteurs de risques. Chaque graphique représente une dimension de facteur de risque. (rouge : premier quartile, bleu : médiane et vert : troisième quartile)



2.3.2.3 *Analyse critique des résultats*

Le Tableau 9 résume les différentes erreurs quadratiques obtenues sur la base de test. En gras est indiqué pour chacun des deux espaces du vecteur d'entrée que l'on a considéré, le meilleur score obtenu.

Table 9.: Synthèse des erreurs quadratiques obtenues par les modèles sur la base de test

modèles	vecteur d'entrée dans $\mathbb{R}^{11}$	vecteur d'entrée dans $\mathbb{R}^{31}$
polynômial	0,0781	**0,000608**
neuronal		
1 couche 20 neurones	0,0587	0,0903
1 couche 50 neurones	0,0926	0,0251
2 couches à 20 neurones	**0,0153**	0,0220

Le modèle polynômial est de très loin le plus performant. Comme nous l'avons remarqué précédemment, on pourrait expliquer cette bonne performance par la nature même des données que nous avons générées. La convexité et la non linéarité de la fonction f n'est peut être pas suffisante pour mettre en avant les capacités prédictives des réseaux de neurones. Néanmoins, dans le cadre d'une utilisation en *curve-fitting* (à savoir estimer la fonction f à partir de l'observation de NAV), les changements de monotonie observés sur les sensibilités (dus à un sur-paramétrage et donc un sur-apprentissage) des réseaux de neurones peuvent nous rendre réticents quant à leur utilisation. En pratique, une étude aussi précise des sensibilités n'est pas possible compte-tenu de la complexité et du temps nécessaire à la production d'une valeur de *Best Estimate*. Néanmoins, il est important d'avoir conscience que malgré une erreur quadratique parfois acceptable, le comportement marginal du réseau de neurone peut différer de la fonction cible qu'il est censé approcher. Dans une optique de détermination de capital économique dans le cadre de Solvabilité 2 ce constat remet en cause la viabilité de la méthode neuronale en *curve-fitting*.

Enfin, les réseaux de neurones testés présentent un sur-paramétrage relativement aux nombres d'observations disponibles dans notre base d'apprentissage. On se retrouve en effet pour les configurations les plus complexes à deux couches, avec 1040 paramètres réels à calibrer sur 150 observations uniquement. Cette disproportion expliquerait ainsi l'allure des courbes des sensibilités obtenues qui laissent présager des phénomènes de sur-apprentissage.

Cependant, obtenir plus de valeurs de f est difficile à obtenir en pratique (Cf le paragraphe sur les Simulations dans les Simulations). Si l'on veut pouvoir comparer les performances des deux méthodes sur des données plus importantes nous devons alors adopter une autre approche comme celle du *Least Squares Monte Carlo*.

2.3.3 *Les réseaux de neurones dans un cadre de Least Squares Monte Carlo*

Nous considérons ici $N \in \mathbb{N}^*$ simulations primaires du même vecteur d'indices action $(S_1, ..., S_{10})$. Pour faire le lien avec les objets mathématiques introduits précédemment,

on effectue plus exactement N tirages aléatoires et indépendants de facteurs de risque $\epsilon = (\epsilon_1, ..., \epsilon_{10})$ à partir desquels nous reconstituons la valeur des indices par formule de Black-Scholes rappelée ci-après. Dans cette partie où l'on adopte une approche de *Least Squares Monte Carlo*, notre fonction cible n'est plus une valeur de portefeuille homogène à une NAV mais le *Cash Flow*[26] actualisé en première période de notre portefeuille à la date $t = 1$ homogène à une VAN.

On rappelle que l'approche consiste à s'appuyer sur le fait que le *Best Estimate* en $t = 1$ (BE_1) peut s'exprimer comme une fonction mesurable dépendant de ϵ :

$$\mathbb{E}^{\mathbb{Q}}(VAN \mid \epsilon) = BE_1 = f(\epsilon) \tag{5}$$

Où $f : \mathbb{R}^d \to \mathbb{R}$ est une fonction mesurable et unique (presque sûrement). En outre, on fait l'hypothèse qu'il existe une variable aléatoire réelle μ tel que $\mathbb{E}^{\mathbb{Q}}(\mu \mid \epsilon) = 0$ et :

$$VAN = f(\epsilon) + \mu.$$

L'objectif est donc de prédire la valeur de VAN en fonction de ϵ d'une part par la méthode polynomiale usuelle en assurance et d'autre part par l'utilisation d'un modèle neuronal. Du point de vue de l'apprentissage statistique cela revient à déterminer un prédicteur optimal à partir de données bruitées.

Ainsi, entre $t = 0$ et $t = 1$, pour tout scénario $i \in [\![1, N]\!]$ et pour tout actif S_k pour $k \in [\![1, 10]\!]$ nous avons :

$$S_{k,i}^{(1)} = e^{[(r - \frac{\sigma_k^2}{2}) + \sigma_k W_{k,i}^{(1)}]}.$$

Afin d'estimer les *Cash Flows* en $t = 1$ (que nous considérons comme des VAN dans notre étude) on simule alors deux trajectoires antithétiques sous la probabilité risque-neutre $\mathbb{Q}$ de sorte à ce qu'on obtienne à l'issue de chaque scénario primaire $i \in [\![1, N]\!]$ et pour tout actif S_k pour $k \in [\![1, 10]\!]$:

$$S_{k,i}^{(2),+} = S_{k,i}^{(1)} e^{[(r - \frac{\sigma_k^2}{2}) + \sigma_k W_{k,i}^{(2)}]}$$

$$S_{k,i}^{(2),-} = S_{k,i}^{(1)} e^{[(r - \frac{\sigma_k^2}{2}) - \sigma_k W_{k,i}^{(2)}]}$$

Pour chaque scénario $i \in [\![1, N]\!]$, notre variable à expliquer est alors :

$$CF_i = \sum_{k=1}^{10} w_k * CashFlow(t = 1, S_{k,i}^{(1)})$$

26 Flux monétaire en français, le terme anglais est néanmoins souvent employé.

avec

$$CashFlow(t=1, S_{k,i}^{(1)}) = e^{-r} * [\frac{1}{2} * (S_{k,i}^{(2),+} - K_k)^+ + \frac{1}{2} * (S_{k,i}^{(2),-} - K_k)^+]$$

où $(x-y)^+ = max(x-y, 0)$.

Afin de comparer les performances du modèle neuronal face au modèle polynomial nous utilisons comme base de test un échantillon de 50 points générés identiquement à la partie précédente. En outre, on souhaite répliquer les pratiques de contrôle qui s'effectuent en compagnie d'assurance lors de l'utilisation de méthode de *Least Squares Monte Carlo* (LSMC). D'une part nous calibrons un modèle sur les valeurs de VAN. D'autre part, afin de contrôler sa pertinence, nous comparons les valeurs fournies par ce modèle avec celles produites par la méthode de référence (car considérée comme la plus précise à ce jour) des Simulations dans les Simulations. Cette dernière méthode ne permettant de produire qu'un nombre faible de points, le modèle calibré à l'aide de la méthode LSMC est testé sur quelques points uniquement. Dans note cas d'étude, les valeurs du portefeuille de *Calls* jouent alors le rôle des points de contrôle homogènes à ceux qu'aurait pu produire une méthode des Simulations dans les Simulations.

Un autre axe de comparaison des modèles sera celui de l'erreur quadratique finale obtenue sur la base d'apprentissage. Cela permet ainsi d'évaluer le comportement des différentes approches face à des données bruitées. Nous espérons ainsi explorer une autre problématique fréquemment rencontrée en apprentissage statistique.

2.3.3.1 *Présentation de la structure neuronale et premiers résultats*

Nous avons focalisé notre comparaison sur la structure neuronale ayant obtenu les meilleures performances dans la partie précédente à savoir la structure à deux couches de 20 neurones. Les fonctions d'activation sont également identiques : celles de la première couche sont des fonctions Tangente Hyperbolique et celles de la deuxième couche des fonctions Sigmoïde.

L'utilisation d'un algorithme de contrôle du gradient a été particulièrement décisive dans la convergence de la méthode de descente de gradient stochastique utilisée pour optimiser nos paramètres. Une divergence de l'algorithme était observée de façon récurrente lorsqu'aucune atténuation de la descente du gradient n'était appliquée. Le nombre d'observations N de notre base d'apprentissage est de 20000. Le nombre d'itérations nécessaires pour calibrer la structure de neurone choisie a été déterminé à 500. Les erreurs quadratiques des modèles sont déterminées sur 50 points identiques à ceux de la base de test de l'approche *curve-fitting*. Ainsi nous pourrons également évaluer l'influence d'un apprentissage sur des points peu bruités. Le Tableau 10 présente les résultats obtenus :

Table 10.: Erreurs quadratiques obtenues sur la base de test après calibrage du modèle par méthode LSMC

modèle	vecteur d'entrée dans $\mathbb{R}^{11}$	vecteur d'entrée dans $\mathbb{R}^{31}$
polynôme	0.082	0.0617
réseau de neurones	0.148	0.0692

On constate que le polynôme présente de nouveau de meilleurs résultats que se soit en considérant les simples facteurs de risque ou des facteurs pris à la puissance 2 et 3. Les résultats obtenus sur nos données simulées ne présentent donc pas d'avantage significatif des réseaux de neurones face à la méthode polynomiale usuelle. Alors qu'une erreur trop faible sur la base d'apprentissage est souvent associée à un phénomène de sur-apprentissage qui se visualise par de mauvaises prédictions sur la base de test, le polynôme présente des bons résultats sur les deux bases. On en déduit que le polynôme présente une meilleure spécification de modèle : l'erreur est plus faible sur la base d'apprentissage (Figure 42) et la prédiction est meilleure sur la base de test.

Figure 42.: Erreur commise par le modèle polynomial sur la base d'apprentissage (le vecteur d'entrée appartient à $\mathbb{R}^{31}$)

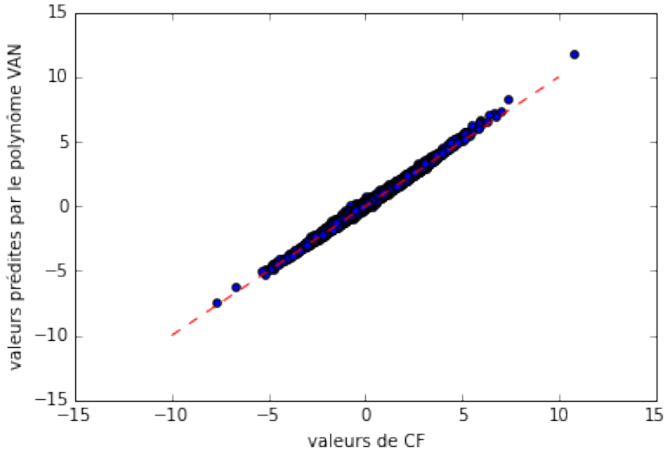
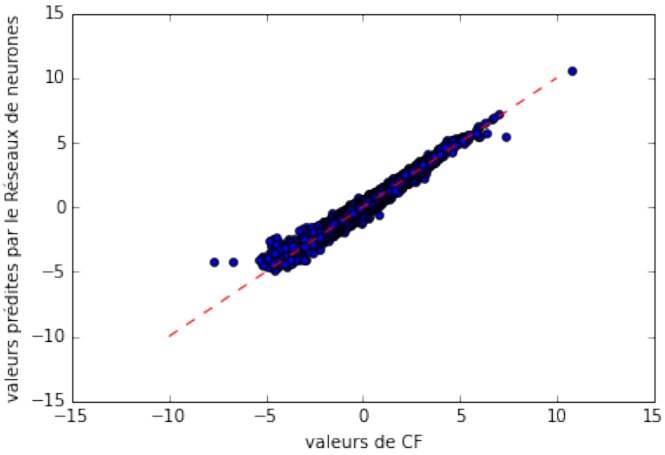
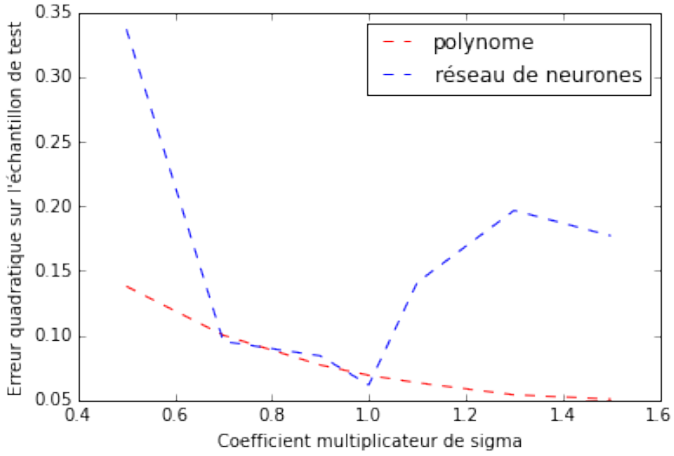


Figure 43.: Erreur commise par le modèle neuronal sur la base d'apprentissage (le vecteur d'entrée appartient à $\mathbb{R}^{31}$ après 500 itérations de calibrage)



Par ailleurs, les données d'apprentissage sont des données bruitées, une trop faible erreur sur la base d'apprentissage est censée impliquer un sur-apprentissage. La Figure 44 permet de comparer l'impact du bruit sur la capacité prédictive du modèle neuronal et du polynôme. Lorsque le bruit est faible (point 0,5 sur la courbe) le réseau de neurones, s'il est sur-paramétré sur-apprend tout comme le polynôme. Ensuite, le réseau de neurones présente des erreurs similaires à celles du polynôme. Lorsque la variance devient trop importante, on remarque que l'erreur commise par le réseau de neurones devient grande alors que celle du polynôme décroît. Cette décroissance confirme de nouveau la bonne spécificité du polynôme au regard des données que nous avons générées.

Figure 44.: Evolution de l'erreur commise sur la base de test par les modèles (entrée dans $\mathbb{R}^{31}$) en fonction de la variance des trajectoires secondaires (CF). La valeur de σ est en variation relative.



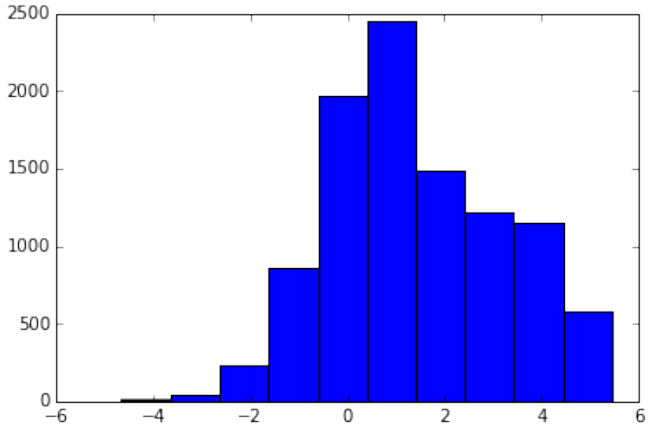
Afin de vérifier que les performances obtenues ne soient pas dépendantes de la nature des données que nous avons simulées, nous avons comparé les deux méthodes sur des données cette fois-ci réelles.

2.3.3.2 *Application à un portefeuille réel d'assurance*

Notre portefeuille réel comporte quatre facteurs de risque financiers [27]. Plus précisément, nous disposons, comme dans notre cas d'étude, de près de 20000 scénarios des facteurs de risque auxquels est associée une valeur de VAN. Ces valeurs ont été générées à l'aide d'un modèle ALM interne à l'entreprise Milliman utilisé par l'équipe de Recherche et Développement pour la génération de capital économique. La distribution des valeurs de VAN est présentée sur l'histogramme de la Figure 45. On dispose également de 9 points de validation générés par la méthode des Simulations dans les Simulations.

27 Les facteurs techniques existants sont constants.

Figure 45.: Distribution des valeurs de VAN



On remarque que la distribution présente des allures similaires à celles que nous avons pu étudier jusqu'à présent : on ne constate pas de symétrie particulière, les valeurs positives apparaissent plus fréquemment. La fonction réelle ayant générée ces valeurs n'est cependant pas connue ici. Notre approche sera donc identique à celle employée dans un contexte opérationnel.

Dans la même démarche que notre cas d'étude, nous avons donc calibré un modèle polynomial. Étant données les corrélations entre les différents facteurs de risque, il a fallu considérer certains termes croisés. Ainsi à partir des quatre facteurs de risque initiaux, nous avons généré 34 facteurs de risque en élevant les monômes à la puissance 2 et 3 puis en considérant différents croisements entre ces termes de degrés 1, 2 et 3.

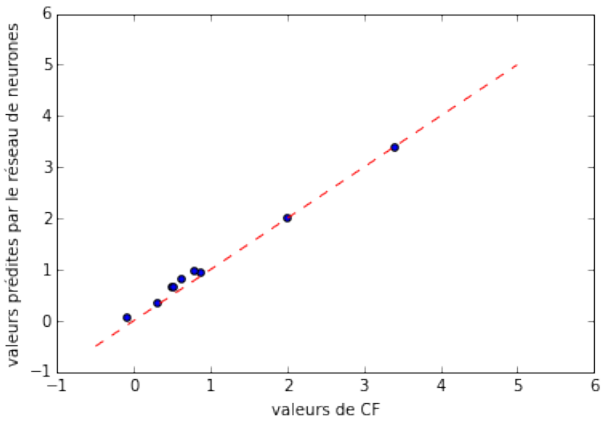
Il est fréquent en assurance de tester différents modèles polynomiaux. Le premier que nous avons calibré est celui tenant compte de l'ensemble des facteurs de risque. D'autres ont été calibrés à l'aide de modèle de sélection *Stepwise* s'appuyant sur des critères d'information statistique AIC ou BIC dont on rappelle la définition en Annexe page 295. Le principe de l'algorithme *Stepwise* est également décrit en Annexe page 297. Le calibrage du polynôme tient compte d'une expertise métier. Au même titre que le choix de la structure du réseau de neurones et de la constitution de ses couches, l'assureur doit être en mesure de choisir les facteurs de risque primaires et ceux générés par transformations polynomiales qu'il intégrera dans son modèle LSMC.

Le modèle neuronal dont nous présentons les résultats est également le modèle neuronal à deux couches de 20 neurones. Les neurones de la première couche ont des fonctions d'activation Tangente Hyperbolique et ceux de la deuxième des fonctions Sigmoïde. Le choix de cette structure a été fait par souci de cohérence avec les analyses précédemment menées. De la littérature sur les réseaux de neurones, il est à noter que le

choix de la structure se fait par empirisme. Plusieurs recherches et tests sont à effectuer avant de pouvoir trouver une structure optimale. Le calibrage a nécessité 300 itérations. La méthode d'optimisation s'est faite par descente de gradient stochastique avec une contrôle de descente de gradient Adagrad.

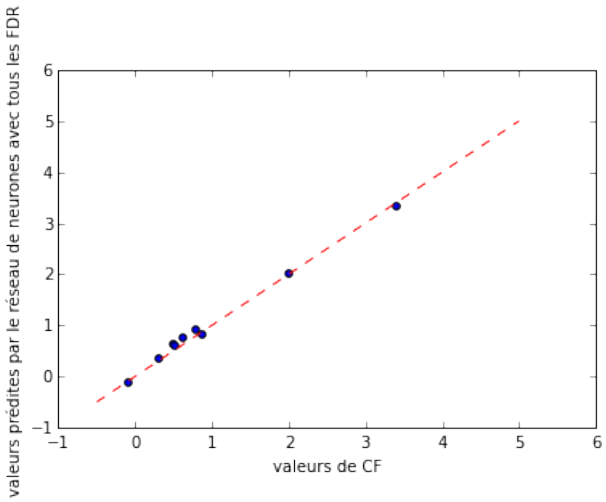
Le premier modèle neuronal a été calibré à partir des 4 facteurs de risque financiers uniquement (aucun terme croisé ni de puissances). La courbe des résultats prédits en fonction des valeurs des points de contrôle (Figure 46) présente une allure acceptable et correcte au regard de celles obtenues sur notre cas d'étude.

Figure 46.: Prédictions du modèle neuronal (vecteur d'entrée dans $\mathbb{R}^5$: 4 facteurs de risque et une constante) en fonction des points de contrôle



Comme l'illustre la Figure 47, la prise en compte de l'ensemble des facteurs de risque (les 34 décrits précédemment) ne semble pas améliorer significativement les prédictions.

Figure 47.: Prédictions du modèle neuronal (vecteur d'entrée dans $\mathbb{R}^{35}$: 34 facteurs de risque et une constante) en fonction des points de contrôle



Les résultats des différents modèles[28] sont résumés dans le Tableau 11 en suivant la nomenclature ci-après :

- OLS : Régression linéaire sur l'ensemble des 34 facteurs de risque (puissances et facteurs croisés);
- SW : Régression linéaire *Stepwise* (seul certains facteurs sont sélectionnés);
- NNET monomes : Réseau de neurones calibré à partir des 4 facteurs de risque financiers initiaux;
- NNET tous les FdR : Réseau de neurones calibré sur les 34 facteurs de risque (puissances et facteurs croisés).

2.3.3.3 *Bilan sur l'utilisation des réseaux de neurones à des fins de détermination de capital économique*

On constate que les prédictions fournies par le réseau de neurones restent raisonnables sur les données réelles. On remarquera que le réseau de neurones ne prenant en compte que les 4 facteurs de risque initiaux prédit des résultats parfois proches des points de contrôle. Par ailleurs, les temps de calibrage des réseaux de neurones restent grâce à la méthode de gradient stochastique raisonnables. Pour les différents calibrages effectués tout au long de cette partie, le temps de calibrage était inférieur à 3 minutes. Ce qui reste raisonnable du point de vue opérationnel. Ce temps peut être considérablement

28 Pour des raisons de confidentialité, les résultats réels ont été transformés.

Table 11.: Résultats selon les points de contrôle fournis par les différents modèles

Points de contrôle	OLS	SW	NNET monomes	NNET tous les FdR
246 561 359	283 741 287	255 705 222	327 190 351	302 801 413
200 188 938	210 284 748	206 714 739	267 091 632	257 907 865
357 259 026	318 905 212	316 809 525	383 265 312	326 452 974
208 330 461	225 853 837	187 020 409	258 871 509	239 504 919
319 892 503	336 927 535	322 722 084	395 569 942	371 539 936
1 417 153 023	1 428 590 425	1 425 842 221	1 408 550 811	1 391 869 482
824 628 324	850 039 770	840 773 722	833 074 421	833 730 385
119 869 630	123 683 122	114 800 756	129 750 661	137 700 854
-48 170 776	-51 399 548	-65 438 948	11 581 574	-64 510 905

réduit par une parallélisation des calculs à laquelle la structure neuronale est adaptée [Sauget 2008] [Zinkevich et al. 2010]. Le temps de calibrage d'un modèle polynomial était quant à lui inférieur à 30 secondes sur la même machine.

Cependant, les modèles polynomiaux présentent une plus grande simplicité pour des résultats plus acceptables pour un assureur. Le modèle polynomial *Stepwise* est en effet le modèle commettant le moins d'erreur de prédiction. Les capacités prédictives des réseaux de neurones semblent alors résulter d'un long travail de recherche sur la structure optimale ainsi que sur le calibrage des méta-paramètres. Dans une optique de détermination de capital économique cette méthode n'est alors que peu concluante. Pour des performances parfois équivalentes, le polynôme reste un modèle paramétrique plus simple à calibrer.

L'étude sur les données réelles confirme ainsi nos premières conclusions. Le modèle polynomial reste plus performant. Cette conclusion reste néanmoins à nuancer car toutes les structures des réseaux de neurones n'ont pas été testées. Ils semblent néanmoins peu judicieux d'établir des structures très complexes si les prédictions sont à peine meilleures.

Enfin, les réseaux de neurones sont apparus comme des modèles moins adaptés que les polynômes pour des calibrages sur des données bruitées. Ce modèle d'apprentissage statistique semble plus sensible, de par son paramétrage complexe, au biais. Le calibrage d'un réseau de neurone est également plus compliqué lorsque la variable de prédiction Y possède une forte variance.

2.3.4 *Étude de l'erreur de modélisation*

Lors de nos recherches, il est apparu que l'échantillonnage avait un impact sur l'erreur commise par notre estimateur. Lorsque nous sommes dans un cadre de *curve-fitting* les

données observées ont une variance moins élevée. En d'autres termes, les observations Y_i (pour reprendre les notations introduites) sont moins bruitées que dans le cadre du *Least Squares Monte Carlo*. Nous avons donc souhaité formaliser ce point.

On considère dans notre problème une variable aléatoire réelle Y de distribution $\mathbb{P}_Y$ que l'on cherche à expliquer.

Soit $d \in \mathbb{N}^*$, soit X un vecteur aléatoire réel de dimension d et de distribution $\mathbb{P}_X$. Dans notre cas d'étude, X représente l'information du marché et Y représente soit la NAV (cas *curve-fitting*) ou soit la VAN (cas LSMC). Le but de ce paragraphe est alors de formaliser l'étude de l'erreur commise en estimant Y.

Nous effectuons pour ce faire une première hypothèse qui est qu'il existe une application mesurable $f : \mathbb{R}^d \to \mathbb{R}$ de sorte que :

$$Y = f(X).$$

On suppose en effet, que la valeur de Y est explicable au travers de l'information de marché X. Afin de pouvoir expliquer et surtout prédire Y à partir de l'observation de X, on cherche alors à déterminer f. Cependant, en pratique, nous ne pouvons observer la vraie valeur de $f(X)$. Dans un cadre de *curve-fitting* cette valeur (qui n'est autre que la NAV) est approchée par des méthodes simulatoires.

Soit $s \in \mathbb{N}^*$ le nombre de simulations effectuées pour estimer f. On suppose qu'il existe une application mesurable $\tilde{f}_s : \mathbb{R}^d \to \mathbb{R}$ et une variable aléatoire réelle η_s de sorte que :

$$f(X) = \tilde{f}_s(X) + \eta_s$$

avec $\eta_s \sim \mathcal{N}(0, \sigma_s^2(X))$. En outre, on pourra écrire $\eta_s = \sigma_s^2(X).\epsilon'$ où $\epsilon' \sim \mathcal{N}(0,1)$ On appellera η_s l'erreur d'échantillonnage et on notera $\mathbb{Q}$ la mesure de probabilité associée.

Toujours pour reprendre l'exemple du *curve-fitting*, cela revient à dire que :

$$NAV(X) = N\tilde{A}V_s(X) + \eta_s$$

où pour une réalisation de X, $N\tilde{A}V_s(X)$ correspond à la valeur obtenue après application d'un modèle ALM sur $s = 1000$ simulations secondaires.

Afin de déterminer notre estimateur on dispose d'une base de données BA de distribution associée $\mathbb{P}_{BA}$

$$BA = \{(\tilde{Y}_1, X_1), ..., (\tilde{Y}_n, X_n)\}$$

où pour tout $i \in [\![1, n]\!]$, $\tilde{Y}_i = \tilde{f}_s(X_i)$.

Il est courant d'effectuer un maillage des données $X_1, ..., X_n$ (c'est-à-dire choisir les scénarios économiques). Ainsi on supposera que $\mathbb{P}_X$ est indépendante de $\mathbb{P}_{BA}$. Ainsi, à partir de cette base d'apprentissage on détermine un estimateur $\hat{f}_{BA}$:

$$\hat{f}_{BA} = \underset{g}{argmin}[\frac{1}{n}\sum_{i=1}^{n}(\tilde{Y}_i - g(X_i))^2].$$

On remarquera que pour $X \sim \mathbb{P}_X$, $\mathbb{E}_{P_{BA}}[\hat{f}_{BA}(X)|X]$ est une variable aléatoire $\mathbb{P}_X$ mesurable. On notera $\mathbb{P} = \mathbb{P}_X \otimes \mathbb{P}_Y \otimes \mathbb{P}_{BA}$ la mesure de probabilité produit sous laquelle l'ensemble des aléas introduits précédemment sont mesurables. Ainsi, nous cherchons à expliciter pour $X \sim \mathbb{P}_X$:

$$\mathbb{E}_{\mathbb{P}_X}[(f(X) - \hat{f}_{BA}(X))^2] = \mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(f(X) - \hat{f}_{BA}(X))^2|X]]$$

ce qui est encore égal à :

$$\mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(f(X) - \tilde{f}_s(X) + \tilde{f}_s(X) - \mathbb{E}_{\mathbb{P}_{BA}}[(\hat{f}_{BA}(X)|X] + \mathbb{E}_{\mathbb{P}_{BA}}[(\hat{f}_{BA}(X)|X] - \hat{f}_{BA}(X))^2]].$$

Or si l'on développe l'expression nous obtenons pour les termes croisés, en utilisant les différentes hypothèses posées précédemment :

$$\begin{aligned}
&\mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(f(X) - \tilde{f}_s(X))(\tilde{f}_s(X) - \mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X])|X] \\
&= \mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(\eta_s) \times (f(X) - \eta_s - \mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X])|X] \\
&= \mathbb{E}_{\mathbb{P}_X}[f(X)\mathbb{E}_{\mathbb{P}}[\eta_s|X] - \mathbb{E}_{\mathbb{P}}[\eta_s^2|X] - \mathbb{E}_{\mathbb{P}}[\eta_s|X]\mathbb{E}_{\mathbb{P}_{\mathbb{BA}}}[\hat{f}_{BA}(X)|X]] \\
&= -\mathbb{E}_{\mathbb{P}_X}[\sigma_s^2(X)]
\end{aligned}$$

$$\begin{aligned}
&\mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(f(X) - \tilde{f}_s(X))(\mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X] - \hat{f}_{BA}(X))|X] \\
&= \mathbb{E}_{\mathbb{P}_X}[(f(X) - \tilde{f}_s(X))(\mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X] - \mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X])] = 0
\end{aligned}$$

$$\mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(\hat{f}_s(X) - \mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X])(\mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X] - \hat{f}_{BA}(X))|X] = 0$$

On obtient finalement :

$$\mathbb{E}_{\mathbb{P}_X}[\mathbb{E}_{\mathbb{P}}[(f(X)-\hat{f}_{BA}(X))^2|X]] = -\mathbb{E}_{\mathbb{P}_X}[\sigma_s^2(X)] + \mathbb{E}_{\mathbb{P}_X}[(\tilde{f}_s(X)-\mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X])^2] + \mathbb{E}_{\mathbb{P}_X}[(\mathbb{E}_{\mathbb{P}_{BA}}[\hat{f}_{BA}(X)|X]-\hat{f}_{BA}(X))^2]$$

L'erreur se décompose alors en 3 termes. Si le premier terme peut au premier abord paraître étrange à cause de son signe, il est néanmoins pertinent. Plus la valeur de s est élevée (cas *curve-fitting*) plus la variance $\sigma_s^2(X)$ augmente et donc l'erreur d'estimation diminue. En d'autres termes, nous disposons de plus d'informations pour estimer la valeur de $f(X) = \mathbb{E}_{\mathbb{Q}}[\tilde{f}_s(X)]$. Il est alors cohérent de commettre moins d'erreur d'estimation de $f(X)$ si notre apprentissage s'effectue sur des données $\tilde{f}_s(X)$ proches des "vraies" valeurs de $f(X)$.

Le deuxième terme représente quant à lui le biais de l'estimateur $\hat{f}_{BA}$ que l'on commet en le définissant sur notre base d'apprentissage BA. Ce biais est d'autant plus faible quand la base de données possède de nombreuse observations (c'est-à-dire lorsque n devient grand).

Enfin, le troisième terme représente la variance de l'estimateur que nous avons construit sur notre base d'apprentissage. Cette quantité tend a augmenter avec la taille de la base d'apprentissage. On retrouve ainsi le dilemme de "biais-variance" présent dans tout problème d'estimation.

2.4 LA MISE EN LUMIÈRE DE NOMBREUSES PROBLÉMATIQUES : DE NOUVEAUX CHAMPS À EXPLORER

Cette première exploration aura permis de mettre en lumière de nombreux sujets et axes qui sont explorés dans les chapitres suivants. Le premier constat est qu'il existe de nombreux points communs entre l'apprentissage statistique et la formalisation traditionnelle des statistiques. Par ailleurs, la régulation semble être un moteur d'innovation et force les entreprises à trouver des solutions concrètes répondant aux exigences réglementaires et opérationnelles. Nous verrons au chapitre 5, que c'est la raison principale qui nous a poussé à étudier l'impact des méthodes d'anonymisation sur les modèles de tarification automobile. Par ailleurs, même s'ils semblent pouvoir augmenter les performances prédictives, les algorithmes de machine learning soulèvent différentes questions quant à la transparence, la parcimonie et la compréhension de ces derniers. C'est la raison pour laquelle le chapitre 4 ainsi que le chapitre 6 se consacrent à ces points précisément.

3

L'ÉCONOMÉTRIE ET LE MACHINE LEARNING

Nos premiers travaux visant à explorer les différents sujets liés à l'utilisation du *machine learning* en assurance nous ont très vite mené à nous interroger sur la frontière qui existe entre les modèles traditionnels et les modèles introduits par la vague du *Big Data*. Si les premiers s'appuient souvent sur l'utilisation de variables exogènes en assurance, ils puisent leurs sources dans les modèles économétriques[1]. Ainsi, nous nous sommes demandés quels étaient précisément les points communs ou différences entre l'économétrie et l'apprentissage statistique. Cette question s'est traduite par la publication d'un article en collaboration avec Arthur Charpentier et Emmanuel Flachaire qui a été publié dans le revue "Economics and Statistics" au premier trimestre 2019 [Charpentier et al. 2019]. Ce chapitre est ainsi la retranscription de cet article.

3.1 INTRODUCTION

L'utilisation de techniques quantitatives en économie remonte probablement au 16ème siècle, comme le montre Morgan (1990). Mais il faudra attendre le début du XXième siècle pour que le terme « économétrie » soit utilisé pour la première fois, donnant naissance à l'*Econometric Society* en 1933. Les techniques d'apprentissage automatique sont plus récentes. C'est à Arthur Samuel, considéré comme le père du premier programme d'auto-apprentissage, que l'on doit le terme « *machine learning* » qu'il définit comme « *a field of study that gives computer the ability without being explicitly programmed* ». Parmi les premières techniques, on peut penser à la théorie des assemblées de neurones proposée dans Hebb (1949) (qui donnera naissance au « *perceptron* » dans les années 1950, puis aux réseaux de neurones) dont Widrow and Hoff (1960) montreront quinze ans plus tard les liens avec les méthodes des moindres carrés, aux SVM (« *support vector machine* ») et plus récemment aux méthodes de *boosting*. Si les deux communautés ont grandi en parallèle, les données massives imposent de créer des passerelles entre les deux approches, en rapprochant les « deux cultures » évoquées par Breiman (2001a), opposant la statistique mathématique (que l'on peut rapprocher de l'économétrie traditionnelle,

1 Lorsqu'ils n'utilisent pas de variables exogènes, les modèles non paramétriques sont souvent probabilistes et proches des modèles des mathématiques financières, que nous ne considérons pas dans ce chapitre.

comme le note Aldrich (2010)) à la statistique computationnelle, et à l'apprentissage machine de manière générale.

3.1.1 *La Modélisation économétrique*

L'économétrie et les techniques d'apprentissage statistique supervisé sont proches, tout en étant très différentes. Proches au départ, car toutes les deux utilisent une base (ou un tableau) de données, c'est à dire des observations $\{(y_i, \boldsymbol{x}_i)\}$, avec $i = 1, \cdots, n$, $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^p$ et $y_i \in \mathcal{Y}$. Si y_i est qualitative, on parlera d'un problème de classification[2], et dans le cas contraire, d'un problème de régression. Proches à l'arrivée, car dans les deux cas, on cherche à construire un « modèle », c'est à dire une fonction $m : \mathcal{X} \mapsto \mathcal{Y}$ qui sera interprétée comme une prévision.

Mais entre le départ et l'arrivée, il existe de réelles différences. Historiquement, les modèles économétriques s'appuient sur une théorie économique, avec le plus souvent des modèles paramétriques. On a alors recours aux outils classiques de l'inférence statistique (comme le maximum de vraisemblance, ou la méthode des moments) pour estimer les valeurs d'un vecteur de paramètres $\boldsymbol{\theta}$, dans un modèle paramétrique $m_{\boldsymbol{\theta}}(\cdot)$, par une valeur $\widehat{\boldsymbol{\theta}}$. Comme en statistique, avoir des estimateurs sans biais est important car on peut quantifier une borne inférieure pour la variance (borne de Cramér-Rao). La théorie asymptotique joue alors un rôle important (développements de Taylor, loi des grands nombres, et théorème central limite). En apprentissage statistique, en revanche, on construit souvent des modèles non-paramétriques, reposant presque exclusivement sur les données (i.e. sans hypothèse de distribution), et les méta-paramètres utilisés (profondeur de l'arbre, paramètre de pénalisation, etc) sont optimisés par validation croisée.

Au delà des fondements, si l'économétrie étudie abondamment les propriétés (souvent asymptotiques) de $\widehat{\boldsymbol{\theta}}$ (vu comme une variable aléatoire, grâce à la représentation stochastique sous-jacente), l'apprentissage statistique s'intéresse davantage aux propriétés du modèle optimal $m^\star(\cdot)$ (suivant un critère qui reste à définir), voire simplement $m^\star(\boldsymbol{x}_i)$ pour quelques observations i jugées d'intérêt (par exemple dans une population de test). Le problème de choix de modèle est aussi vu sous un angle assez différent. Suivant la loi de Goodhart (« *si une mesure devient un objectif, elle cesse d'être une mesure* »), les économètres utilisent des critères de type AIC ou BIC pour choisir un modèle optimal (pénalisant la qualité d'ajustement d'un modèle par sa complexité, ex-post, lors de la

2 Nous utiliserons ici le terme « classification » lorsque $\mathcal{Y}$ est un ensemble de classes, typiquement une classification binaire, $\mathcal{Y} = \{0, 1\}$, ce cas correspondant à la réalisation d'une variable indicatrice, $\mathbf{1}_{Y_t \leq 0}$, ou $\mathbf{1}_{Y \in \mathcal{A}}$, par exemple. Ce terme est moins daté que « discrimination » par exemple, et plus général que la constitution d'un « score » (qui est souvent une étape intermédiaire). Il ne doit pas être confondu avec la classification non-supervisée (comme la « classification ascendante hiérarchique ») qui est la constitution de classe homogène à partir d'une mesure de similarité (on utilisera parfois, dans ce cas, le terme de « constitution de classes », ou de « clusters »).

phase de validation ou de choix), alors qu'en apprentissage statistique, c'est la fonction objectif qui tiendra compte d'une pénalisation, comme pour le LASSO, ressemblant à une forme de pénalisation ex-ante.

3.1.2 *Applications*

Avant de revenir sommairement sur l'évolution des modèles économétriques, c'est à Francis Galton que l'on doit le terme « régression », comme le rappelle Koenker et al. (1998). Si le terme est parfois devenu synonyme de « modèle économétrique », il avait été introduit dans le contexte de « *regression towards mediocraty inhereditary stature* », pour reprendre le titre de l'article paru en 1886. Galton utilisait un modèle linéaire pour modéliser la taille moyenne d'un garçon (à l'âge adulte) en fonction de la taille de son père. Si cette technique de régression était connue par les économistes, il a fallu attendre les années 1930 pour voir surgir le concept de « modèle » économique. Comme le note Debreu (1986), la première étape a été de formuler des affirmations économiques dans un langage mathématique. Les différentes grandeurs sont vues comme des variables, et dans les années 1930, on verra apparaître les « *statistical demand curves* », pour reprendre la terminologie d'Henry Schultz. Cette approche statistique permettra d'aller plus loin que les travaux pionniers de Engel (1857) qui étudiait empiriquement la relation entre la consommation et le revenu des ménages, par exemple, dans une approche uniquement descriptive.

Les modèles économétriques se sont développés en parallèle des modèles macro-économiques. Les premiers travaux de la Commission Cowles ont porté sur l'identification des modèles économiques, et l'estimation de modèles à équations simultanées. Ces développements vont aboutir à un âge d'or de l'économétrie, dans les années 1960, où les modèles économétriques seront utilisés afin d'améliorer les prévisions macro-économiques. On va alors voir apparaître tout un ensemble de « lois » qui sont souvent traduites comme des relations linéaires entre des grandeurs agrégées, telle que la « loi de Okun » introduite dans Okun (1962) qui postule une relation linéaire entre le variation du nombre de demandeurs d'emploi et de la croissance du PIB,

$$\Delta\text{Chômage}_t = \beta_0 + \beta_1 \text{Croissance}_t + \varepsilon_t,$$

quand on étudie ces grandeurs au cours du temps (t), ou la loi de « Feldstein-Horioka » introduite dans Feldstein and Horioka (1857) qui suppose une relation linéaire entre les taux d'investissement et d'épargne, relativement au revenu national,

$$\frac{\text{investissement}_i}{\text{revenu national}_i} = \beta_0 + \beta_1 \frac{\text{épargne}_i}{\text{revenu national}_i} + \varepsilon_i$$

quand on modèlise les liens entre les allocations investissement-épargne pour plusieurs pays (i). Cet âge d'or correspond aussi à un questionnement profond, suite à la critique

de Lucas (1976), s'interrogeant sur l'inefficacité de ces outils à expliquer et à prévoir des crises. La principale explication était alors le manque de fondement micro-économiques de ces modèles, ce qui donnera un second souffle aux modèles micro-économétriques. On pourra rappeler que cette critique dite « de Lucas » avait été formulée dans Orcutt (1952), qui avançait l'idée que les données macro-économiques posaient des problèmes insolubles d'identification. La solution passait forcément par de l'économétrie sur données individuelles (au lieu de données obtenues par agrégation), ce qui sera reformulé quelques années plus tard par Koopmans (1957).

Malheureusement, les modèles micro-économétriques sont généralement plus complexes, car ils se doivent de tenir compte d'une éventuelle censure dans les données, avec par exemple le modèle introduit par Tobin (1958), d'erreurs sur les variables (qui pourront être corrigées par des instruments avec une technique initiée par Reiersø l (1945)) ou avoir été collectées avec un biais de sélection, avec les techniques proposées par Heckman (1979). On notera que les économètres se sont beaucoup interrogés sur la manière dont les données étaient construites, et ne se sont jamais contentés de « construire des modèles ». Un exemple peut être l'évaluation des politiques publiques, largement détaillé dans Givord (2010). Dans ce cas, en effet, deux écoles se sont opposées (initiant un débat que l'on verra resurgir tout au long de l'article sur les méthodes d'apprentissage statistique). La première, dite « structuraliste », cherchera à construire un modèle complet afin de décrire le comportement des agents économiques. La seconde, souvent qualifiée d'« empiriste », vise à tester l'effet d'une mesure sans pour autant expliciter les mécanismes sous-jacents. C'est ce qu'explique Angrist and Krueger (1991), en affirmant « *research in a structuralist style relies heavily on economic theory to guide empirical work* [· · ·] *An alternative to structural modeling,* [· · ·] *the 'experimentalist' approach,* [· · ·] *puts front and center the problem of identifying causal effects from specific events or situations* ».

On peut aussi souligner que si l'approche de la Commission Cowles était très exigeante, en supposant le modèle connu, toute une littérature s'est développée en allégeant cette hypothèse, soit en essayant de choisir le bon modèle (avec les travaux de Hendry and Krolzig (2001) par exemple) ou en proposant de faire des moyennes de modèles (comme développé récemment par Li and Zhang (2017)). Et plus généralement, alors que l'analyse économétrique (en particulier à des fins de politique économique) s'est développée plus récemment autour de l'inférence causale, les techniques d'apprentissage machine ont été vues, traditionnellement, autour de la prédiction (où la recherche de corrélations suffisamment fortes entre variables suffit) d'où leur popularité dans des usages plus industriels de classification, comme la reconnaissance de caractères, de signature, d'images, ou de traduction, comme le rappelle Bishop (2006). En biologie, ces techniques ont été appliquées pour créer des classifications d'espèces animales en fonction d'analyse d'ADN, ou dans le domaine militaire et sécuritaire pour l'identification de cibles ou de terroristes (potentiels). Il faudra attendre les années 1990 pour voir des applications en finance avec Altman and Varetto (1994) par exemple, ou Herbrich and Obermayer (1999) pour une revue de littérature sur les applications potentielles en économie. Si des

applications sont aujourd'hui nombreuses, et si ces techniques concurrencent les modèles de micro-économétrie (on pourra penser au « scoring » bancaire, à la détection de fraude fiscale ou assurantielle, ou à l'identification de prospects en marketing), les algorithmes d'apprentissage sont devenus très populaires en reconnaissance de parole, puis d'images, et plus récemment avec les applications en ligne et les applications aux jeux (d'échec, et plus récemment de go). Si l'économétrie s'est développée au confluent des mathématiques et de l'économie, l'apprentissage machine (que l'on pourrait avoir tendance à rapprocher de l'intelligence artificielle) s'est développé à la frontière des mathématiques et de l'informatique (avec des résultats fondamentaux en optimisation - en particulier autour des méthodes de gradient stochastique - et sur les espaces « sparse » ou « parcimonieux »).

3.1.3 *De la grande dimension aux données massives*

Dans cet article, une variable sera un vecteur de $\mathbb{R}^n$, de telle sorte qu'en concaténant les variables ensemble, on puisse les stocker dans une matrice $\boldsymbol{X}$, de taille $n \times p$, avec n et p potentiellement grands[3]. Le fait que n soit grand n'est, a priori, pas un problème en soi, au contraire. De nombreux théorèmes en économétrie et en statistique sont obtenus lorsque $n \to \infty$ (c'est la théorie asymptotique). En revanche, le fait que p soit grand est problématique, en particulier si $p > n$. Les deux dimensions sont à distinguer, car elles vont engendrer des problèmes relativement différents.

Portnoy (1988) a montré que l'estimateur du maximum de vraisemblance conserve la propriété de normalité asymptotique si p reste petit devant n, ou plus précisément, si $p^2/n \to 0$ lorsque $n, p \to \infty$. Aussi, il n'est pas rare de parler de grande dimension dès lors que $p > \sqrt{n}$. Un autre concept important est celui de sparsité, qui repose non pas sur la dimension p mais sur la dimension effective, autrement dit le nombre de variables effectivement significatives. Il est alors possible d'avoir $p > n$ tout en ayant des estimateurs convergents.

La grande dimension en terme de nombre de variables, p, peut faire peur à cause de la malédiction de la dimension, introduit par Bellman (1957). L'explication de cette malédiction est que le volume de la sphère unité, en dimension p, tend vers 0 lorsque $p \to \infty$. On dit alors que l'espace est « parcimonieux » - c'est à dire que la probabilité de trouver un point proche d'un autre devient de plus en plus faible (on pourrait parler d'espace « clairsemé »). Ou de manière duale, pour reprendre la formulation de Hastie and Friedman (2009), le volume qu'il convient de considérer pour avoir une proportion donnée d'observations augmente avec p. L'idée de réduire la dimension en considérant

3 Là encore, des extensions sont possibles, en particulier dans les données médicales avec des images de type IRM comme variables prédictives, ou des données climatiques avec des cartes en variables prédictives, ou plus généralement une variable tensorielle en dimension plus ou moins grande. Comme le montre Kolda and Bader (2009) il est toutefois possible de se ramener dans le cas usuel (de données sous formes de vecteurs) en utilisant la décomposition de Tucker.

une analyse en composante principale peut paraître séduisante, mais l'analyse souffre d'un certain nombre de défauts en grande dimension. La solution est alors souvent la sélection de variables, qui pose le problème des tests multiples, ou du temps de calcul, pour sélectionner k variables parmi p, lorsque p est grand.

Pour reprendre la terminologie de Bühlmann and van de Geer (2011), les problèmes que nous évoquons ici correspondent à ceux observés en grande dimension, qui est un problème essentiellement statistique. D'un point de vue informatique, on peut aller un peu plus loin, avec des données réellement massives (qui occupent énormément de place en mémoire). Dans ce qui précède, les données étaient stockées dans une matrice $\boldsymbol{X}$, de taille $n \times p$. Si cet objet formel est toujours bien défini, il peut y avoir des soucis à stocker une telle matrice, voire manipuler une matrice abondamment utilisée en économétrie, $\boldsymbol{X}^\top\boldsymbol{X}$ (matrice $n \times n$). La condition du premier ordre (dans le modèle linéaire) est associée à la résolution de $\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) = \boldsymbol{0}$. En dimension raisonnable, on utilise la décomposition QR (c'est à dire la décomposition de Gram-Schmidt). En grande dimension, on peut utiliser des méthodes numériques de descente de gradient, où le gradient est approché sur un sous-échantillon de données (comme décrit par exemple dans Zinkevich et al. (2001)) Cet aspect informatique est souvent oublié alors qu'il a été à la base de bon nombre d'avancées méthodologiques, en économétrie. Par exemple, Hoerl and Kennard (1981) reviennent sur l'origine de l'utilisation de la régression Ridge: « *Nous facturions 90$ par jour pour notre temps, mais avons dû facturer 450$ par heure d'ordinateur sur un Univac I* $(\cdots)$ *Avec cette machine, il a fallu 75 minutes de traitement pour inverser une matrice* 40×40 *en passant par une partition* 4×4 *de sous-matrices* 10×10*, en utilisant des bandes magnétiques pour le stockage temporaire. Nous avons noté que les coefficients d'un régression linéaire calculés en utilisant les moindres carrés n'avaient pas toujours de sens. Les coefficients avaient tendance à être trop grands en valeur absolue, certains avaient même le mauvais signe, et ils pouvaient être instables avec de très petits changements dans les données* $(\cdots)$ *Comme la méthode que nous proposions attaquait l'une des vaches sacrées de la régression linéaire - les moindres carrés - nous avons fait face à une une résistance considérable*».

3.1.4 *Statistique computationnelle et non-paramétrique*

L'objet de ce papier est d'expliquer les différences majeures entre l'économétrie et l'apprentissage statistique, correspondant aux deux cultures mentionnées par Breiman (2001a), lorsqu'il évoque en modélisation statistique la « *data modeling culture* » (reposant sur un modèle stochastique, comme la régression logistique ou le modèle de Cox) et la « *algorithmic modeling culture* » (reposant sur la mise en œuvre d'un algorithme, comme dans les forêts aléatoires ou les supports vecteurs machines, une liste exhaustive est présenté dans Shalev-Shwartz and Ben-David (2014)). Mais la frontière entre les deux est très poreuse. à l'intersection se retrouve, par exemple, l'économétrie non-paramétrique. Cette dernière

repose sur un modèle probabiliste (comme l'économétrie), tout en insistant davantage sur les algorithmes (et leurs performances) plutôt que sur des théorèmes asymptotiques.

L'économétrie non-paramétrique repose sur des décompositions dans des bases fonctionnelles. L'économétrie linéaire consiste à approcher la fonction $m : \boldsymbol{x} \mapsto \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$ par une fonction linéaire. Mais plus généralement, on peut considérer une décomposition dans une base fonctionnelle, et s'intéresser à une approximation obtenue sur un nombre fini de termes :

$$m(\boldsymbol{x}) = \sum_{j=0}^{\infty} \omega_j g_j(\boldsymbol{x}) \quad \text{et} \quad \widehat{m}(\boldsymbol{x}) = \sum_{j=0}^{h^\star} \widehat{\omega}_j g_j(\boldsymbol{x}),$$

où les poids ω_j sont estimés, alors que le nombre de composantes $h^\star$ est optimisé. On retrouvera ici les modèles additifs (dits GAM), par exemple, étudiés dans Hastie and Tibshirani (1990). Une autre solution consiste à considérer un modèle simple, mais local. Par exemple un modèle constant, au voisinage de $\boldsymbol{x}$, obtenu en considérant seulement les observations proches de $\boldsymbol{x}$:

$$\widehat{g}(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\omega}_{\boldsymbol{x}} y_i \quad \text{par exemple} \quad \widehat{g}(\boldsymbol{x}) = \frac{1}{n_{\boldsymbol{x}}} \sum_{i:\|\boldsymbol{x}_i - \boldsymbol{x}\| \leq h} y_i$$

où $n_{\boldsymbol{x}}$ est le nombre d'observations au voisinage de $\boldsymbol{x}$. En mettant des poids fonctions de la distance à $\boldsymbol{x}$, on retrouve ici le modèle obtenu par Nadaraya (1964) et Watson (1964), ou les méthodes de régression locale.

Les différentes méthodes reposent sur des méta-paramètres - correspondant paramètres de lissage - c'est à dire h dans les exemples précédents. Pour un économètre, le paramètre « optimal » pour h est obtenu soit à l'aide de théorèmes asymptotiques, soit à l'aide de techniques de validation, comme en apprentissage automatique. On obtient alors une valeur numérique, mais on n'a pas d'interprétation en lien avec la taille de l'échantillon, ou les variances des différentes grandeurs. Si les économistes ont toujours la culture du tableau présentant la « sortie de régression », les méthodes non-paramétriques sont utiles pour détecter des mauvaises spécifications, des non-prises en compte de nonlinéarité, ou d'effets croisées (et les outils d'apprentissage automatique que nous allons voir peuvent probablement jouer le même rôle).

3.1.5 *Plan du chapitre*

Pour reprendre le titre de Varian (2014), l'objet de l'article restitué dans ce chapitre est de présenter les différences fondamentales entre l'économétrie et l'apprentissage machine, et surtout de voir comment ces deux techniques peuvent apprendre l'une de l'autre, dans un contexte où les bases de données deviennent massives. La Section 3.2 reviendra sur la

construction du modèle linéaire. Le modèle sera introduit ici à partir du modèle Gaussien « homoscédastique ». Ce modèle présente l'avantage d'avoir une élégante interprétation géométrique, en terme de projection sur le sous-espace des combinaisons linéaires des variables explicatives. La première extension que nous verrons est le passage du modèle linéaire à un modèle non-linéaire, tout en construisant un prédicteur linéaire. La seconde extension proposera de construire un modèle non-gaussien, pour modéliser une variable indicatrice ou un comptage Poissonnien, par exemple, donnant naissance aux modèles linéaires généralisés (construits pour des variables dans la famille exponentielle).

Une fois rappelée l'origine des outils économétriques standards, dans la Section 3.3 nous présenterons les outils et techniques développés dans le contexte de l'apprentissage machine. Si l'outil central des modèles économétriques est la distribution de la variable dépendante, Y, les techniques d'apprentissage reposent sur une fonction de perte, ℓ, représentant une « distance » entre la variable d'intérêt y, et le modèle $m(\cdot)$. Nous présenterons tout d'abord l'algorithme de boosting, reposant sur l'idée d'un apprentissage lent, en modélisant séquentiellement les résidus. Le danger des méthodes d'apprentissage est qu'il est aisé de construire un modèle « parfait », dont le pouvoir prédictif serait faible. Nous évoquerons alors les techniques de pénalisation, utilisées pour éviter le surapprentissage. Nous évoquerons en particulier les notions d'*in-sample* et *out-of-sample*, et les techniques de validation croisée. Pour conclure cette section, nous reviendrons sur les interprétations probabilistes des outils d'apprentissage, qui permettront de faire le lien entre les différentes approches, tout en restant sur une discussion générale sur la philosophie de ces deux cultures.

Après cette section sur la philosophie des méthodes d'apprentissage automatique, nous reviendrons dans la section 3.4 sur quelques algorithmes importants : les réseaux de neurones, les supports vecteurs machine (SVM) et enfin les méthodes de type arbres et forêts.

La Section 3.5 proposera des exemples concrets de comparaison entre les différentes techniques, dans le cas de classifications (binaires) pour des variables $y \in \{0, 1\}$ (achat d'assurance, non-remboursement d'un crédit) et dans un contexte de régression (lorsque la variable d'intérêt n'est plus qualitative - ce que nous simplifierons en notant $y \in \mathbb{R}$). Nous reviendrons avant sur les courbes ROC, outils importants pour juger de la qualité d'un classifieur, malheureusement peu utilisés en économétrie.

Nous verrons en particulier les méthodes de « bagging », forêts aléatoires ou « boosting ». Nous reviendrons aussi sur les méthodes de choix de modèles et des méta-paramètres. à travers ces exemples d'application, nous verrons comment les modèles de type apprentissage automatique peuvent être utilisés pour mieux détecter la mauvaise spécification des modèles de régression paramétriques, à cause de non-linéarités, et/ou d'interactions manquées.

3.2 ÉCONOMÉTRIE ET MODÈLE PROBABILISTE

L'importance des modèles probabilistes en économie trouve sa source dans les questionnements de Working (1927) et les tentatives de réponses apportées dans les deux tomes de Tinbergen (1939). Ces derniers ont engendré par la suite énormément de travaux, comme le rappelle Duo (1993) dans son ouvrage sur les fondements de l'économétrie, et plus particulièrement dans le premier chapitre « *The Probability Foundations of Econometrics* ». Rappelons que Trygve Haavelmo a reçu le prix Nobel d'économie en 1989 pour sa « *clarification des fondations de la théorie probabiliste de l'économétrie* ». Car comme l'a montré Haavelmo (1944) (initiant un changement profond dans la théorie économétrique dans les années 1930, comme le rappelle le chapitre 8 de Morgan (1990)) l'économétrie repose fondamentalement sur un modèle probabiliste, et ceci pour deux raisons essentielles. Premièrement, l'utilisation de grandeurs (ou « mesures ») statistiques telles que les moyennes, les erreurs-types et les coefficients de corrélation à des fins inférentielles ne peut se justifier que si le processus générant les données peut être exprimé en termes de modèle probabiliste. Deuxièmement, l'approche par les probabilités est relativement générale, et se trouve être particulièrement adaptée à l'analyse des observations « dépendantes » et « non homogènes », telles qu'on les trouve souvent sur des données économiques. On va alors supposer qu'il existe un espace probabiliste $(\Omega, \mathcal{F}, \mathbb{P})$ tel que les observations $(y_i, \boldsymbol{x}_i)$ sont vues comme des réalisations de variables aléatoires $(Y_i, \boldsymbol{X}_i)$. En pratique, la loi jointe du couple $(Y, \boldsymbol{X})$ nous intéresse toutefois peu : la loi de $\boldsymbol{X}$ est inconnue, et c'est la loi de Y conditionnelle à $\boldsymbol{X}$ qui nous intéressera. Dans la suite, nous noterons x une observation, $\boldsymbol{x}$ un vecteur d'observations, X une variable aléatoire, et $\boldsymbol{X}$ un vecteur aléatoire et, abusivement, $\boldsymbol{X}$ pourra aussi désigner la matrice des observations individuelles (les $\boldsymbol{x}_i$), suivant le contexte.

3.2.1 *Fondements de la statistique mathématique*

Comme le rappelle l'introduction de Vapnik (1998), l'inférence en statistique paramétrique est basée sur la croyance suivante: le statisticien connaît bien le problème à analyser, en particulier, il connaît la loi physique qui génère les propriétés stochastiques des données, et la fonction à trouver s'écrit via un nombre fini de paramètres[4]. Pour trouver ces paramètres, on adopte la méthode du maximum de vraisemblance. Le but de la théorie est de justifier cette approche (en découvrant et en décrivant ses propriétés favorables). On verra qu'en apprentissage, la philosophie est très différente, puisqu'on ne dispose pas d'informations a priori fiables sur la loi statistique sous-jacente au problème, ni-même sur la fonction que l'on voudrait approcher (on va alors proposer des méthodes pour construire une approximation à partir de données à notre disposition, pour reprendre Vapnik (1998)). Un « âge d'or » de l'inférence paramétrique, de 1930 à 1960, a posé les

4 On peut rapprocher cette approche de l'économétrie structurelle, telle que présentée par exemple dans Kean (2010).

bases de la statistique mathématique, que l'on retrouve dans tous les manuels de statistique, y compris aujourd'hui. Comme le dit Vapnik (1998), le paradigme paramétrique classique est basé sur les trois croyances suivantes:

1. Pour trouver une relation fonctionnelle à partir des données, le statisticien est capable de définir un ensemble de fonctions, linéaires dans leurs paramètres, qui contiennent une bonne approximation de la fonction souhaitée. Le nombre de paramètres décrivant cet ensemble est petit.

2. La loi statistique sous-jacente à la composante stochastique de la plupart des problèmes de la vie réelle est la loi normale. Cette croyance a été soutenue en se référant au théorème de limite centrale, qui stipule que dans de larges conditions la somme d'un grand nombre de variables aléatoires est approximée par la loi normale.

3. La méthode du maximum de vraisemblance est un bon outil pour estimer les paramètres.

Nous reviendrons dans cette partie sur la construction du paradigme économétrique, directement inspiré de celui de la statistique inférentielle classique.

3.2.2 *Lois conditionnelles et vraisemblance*

L'économétrie linéaire a été construite sous l'hypothèse de données individuelles, ce qui revient à supposer les variables $(Y_i, \boldsymbol{X}_i)$ indépendantes (s'il est possible d'imaginer des observations temporelles - on aurait alors un processus $(Y_t, \boldsymbol{X}_t)$ - mais nous n'aborderons pas les séries temporelles dans cet article). Plus précisément, on va supposer que conditionnellement aux variables explicatives $\boldsymbol{X}_i$, les variables Y_i sont indépendantes. On va également supposer que ces lois conditionnelles restent dans la même famille paramétrique, mais que le paramètre est une fonction de $\boldsymbol{x}$. Dans le modèle linéaire Gaussien on suppose que :

$$(Y|\boldsymbol{X}=\boldsymbol{x}) \overset{\mathcal{L}}{\sim} \mathcal{N}(\mu(\boldsymbol{x}), \sigma^2) \quad \text{avec} \quad \mu(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}, \text{ et } \boldsymbol{\beta} \in \mathbb{R}^p. \tag{6}$$

On parle de modèle linéaire car $\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] = \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}$ est une combinaison linéaire des variables explicatives. C'est un modèle homoscédastique si $\text{Var}[Y|\boldsymbol{X}=\boldsymbol{x}] = \sigma^2$, où σ^2 est une constante positive. Pour estimer les paramètres, l'approche classique consiste à utiliser l'estimateur du Maximum de Vraisemblance, comme l'avait suggéré initialement Ronald Fisher. Dans le cas du modèle linéaire Gaussien, la log-vraisemblance s'écrit :

$$\log \mathcal{L}(\beta_0, \boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, \boldsymbol{x}) = -\frac{n}{2}\log[2\pi\sigma^2] - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})^2.$$

Notons que le terme de droite, mesurant une distance entre les données et le modèle, va s'interpréter comme la déviance, dans les modèles linéaires généralisés. On va alors poser :

$$(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2) = \text{argmax}\Big\{ \log \mathcal{L}(\beta_0, \boldsymbol{\beta}, \sigma^2 | \boldsymbol{y}, \boldsymbol{x}) \Big\}.$$

L'estimateur du maximum de vraisemblance est obtenu par minimisation de la somme des carrés des erreurs (estimateur dit des « moindres carrés ») que nous retrouverons dans l'approche par apprentissage automatique.

Les conditions du premier ordre permettent de retrouver les équations normales, dont l'écriture matricielle est $\boldsymbol{X}^\top[\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}] = \boldsymbol{0}$, que l'on peut aussi écrire $(\boldsymbol{X}^\top\boldsymbol{X})\widehat{\boldsymbol{\beta}} = \boldsymbol{X}^\top\boldsymbol{y}$. Si la matrice $\boldsymbol{X}$ est de plein rang colonne, alors on retrouve l'estimateur classique :

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{\beta} + (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{\varepsilon} \tag{7}$$

en utilisant une écriture basée sur les résidus (comme souvent en économétrie), $y = \boldsymbol{x}^\top\boldsymbol{\beta} + \varepsilon$. Le théorème de Gauss Markov assure que cette estimateur est l'estimateur linéaire sans biais de variance minimale. On peut alors montrer que $\widehat{\boldsymbol{\beta}} \overset{\mathcal{L}}{\sim} \mathcal{N}(\boldsymbol{\beta}, \sigma^2[\boldsymbol{X}^\top\boldsymbol{X}]^{-1})$, et en particulier :

$$\mathbb{E}[\widehat{\boldsymbol{\beta}}] = \boldsymbol{\beta} \quad \text{et} \quad \text{Var}[\widehat{\boldsymbol{\beta}}] = \sigma^2[\boldsymbol{X}^\top\boldsymbol{X}]^{-1}.$$

En fait, l'hypothèse de normalité permet de faire un lien avec la statistique mathématique, mais il est possible de construire cet estimateur donné par l'équation (7). Si on suppose que $Y|\boldsymbol{X} = \boldsymbol{x} \overset{\mathcal{L}}{\sim} \boldsymbol{x}^\top\boldsymbol{\beta} + \varepsilon$, avec $\mathbb{E}[\varepsilon] = 0$, $\text{Var}[\varepsilon] = \sigma^2$, $\text{Cov}[\varepsilon, X_j] = 0$ pour tout j, alors $\widehat{\boldsymbol{\beta}}$ est un estimateur sans biais de $\boldsymbol{\beta}$ ($\mathbb{E}[\widehat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$) et de variance minimale parmi les estimateurs sans biais linéaires, avec $\text{Var}[\widehat{\boldsymbol{\beta}}] = \sigma^2[\boldsymbol{X}^\top\boldsymbol{X}]^{-1}$. De plus, cet estimateur est asymptotiquement normal

$$\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \overset{\mathcal{L}}{\to} \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \quad \text{lorsque} \quad n \to \infty$$

La condition d'avoir une matrice $\boldsymbol{X}$ de plein rang peut être (numériquement) forte en grande dimension. Si elle n'est pas vérifiée, $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$ n'existe pas. Si $\mathbb{I}$ désigne la matrice identité, notons toutefois que $(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$ existe toujours, pour $\lambda > 0$. Cet estimateur est appelé l'estimateur RIDGE de niveau λ (introduit dans les années 60 par Hoerl (1962), et associé à une régularisation étudiée par Tikhonov (1963)). Cette estimateur apparaît naturellement dans un contexte d'économétrie Bayesienne (nous le reverrons dans la section suivante présentant les techniques d'apprentissage automatique).

3.2.3 *Les résidus*

Il n'est pas rare d'introduire le modèle linéaire à partir de la loi des résidus, comme nous l'avions mentionné auparavant. Aussi, l'équation (6) s'écrit aussi souvent :

$$y_i = \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} + \varepsilon_i \tag{8}$$

où les ε_i sont des réalisations de variables aléatoires i.i.d., de loi $\mathcal{N}(0, \sigma^2)$. On notera parfois $\boldsymbol{\varepsilon} \overset{\mathcal{L}}{\sim} \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I})$, sous une forme vectorielle. Les résidus estimés sont définis par :

$$\widehat{\varepsilon}_i = y_i - \left[\widehat{\beta}_0 + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}\right]$$

Ces résidus sont l'outil de base pour diagnostiquer la pertinence du modèle.

Une extension du modèle décrit par l'équation (6) a été proposé pour tenir compte d'un éventuel caractère hétéroscédastique :

$$(Y|\boldsymbol{X} = \boldsymbol{x}) \overset{\mathcal{L}}{\sim} \mathcal{N}(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x}))$$

où $\sigma^2(\boldsymbol{x})$ est une fonction positive des variables explicatives. On peut réécrire ce modèle en posant :

$$y_i = \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} + \sigma^2(\boldsymbol{x}_i) \cdot \varepsilon_i$$

où les résidus sont toujours i.i.d., mais de variance unitaire,

$$\varepsilon_i = \frac{y_i - [\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}]}{\sigma(\boldsymbol{x}_i)}.$$

Si l'écriture à l'aide des résidus est populaire en économétrie linéaire (lorsque la variable dépendante est continue), elle ne l'est toutefois plus dans les modèles de comptage, ou la régression logistique.

L'écriture à l'aide d'un terme d'erreur (comme dans l'équation (8)) pose toutefois de nombreuses questions quant à la représentation d'une relation économique entre deux grandeurs. Par exemple, on peut supposer qu'il existe une relation (linéaire pour commencer) entre les quantités d'un bien échangé, q et son prix p. On peut ainsi imaginer une équation d'offre

$$q_i = \beta_0 + \beta_1 p_i + u_i$$

(u_i désignant un terme d'erreur) où la quantité vendue dépend du prix, mais de manière tout aussi légitime, on peut imaginer que le prix dépend de la quantité produite (ce qu'on pourrait appeler une équation de demande),

$$p_i = \alpha_0 + \alpha_1 q_i + v_i$$

(v_i désignant un autre terme d'erreur). Historiquement, le terme d'erreur dans l'équation (8) a pu être interprété comme une erreur idiosyncratique sur la variable y, les variables dites explicatives étant supposées fixées, mais cette interprétation rend souvent le lien entre une relation économique et un modèle économique compliqué, la théorie économique parlant de manière abstraite d'une relation entre grandeur, la modélisation économétrique imposant une forme spécifique (quelle grandeur est y et quelle grandeur est x) comme le montre plus en détails le chapitre 7 de Morgan (1990).

3.2.4 *Géométrie du modèle linéaire Gaussien*

Définissons le produit scalaire dans $\mathbb{R}^n$, $\langle \boldsymbol{a}, \boldsymbol{b}\rangle = \boldsymbol{a}^\top \boldsymbol{b}$, et notons $\|\cdot\|$ la norme euclidienne associée, $\|\boldsymbol{a}\| = \sqrt{\boldsymbol{a}^\top \boldsymbol{a}}$ (notée $\|\cdot\|_{\ell_2}$ dans la suite). Notons $\mathcal{E}_{\boldsymbol{X}}$ l'espace engendré par l'ensemble des combinaisons linéaires des composantes $\boldsymbol{x}$ (en rajoutant la constante). Si les variables explicatives sont linéairement indépendantes, $\boldsymbol{X}$ est de plein rang colonne et $\mathcal{E}_{\boldsymbol{X}}$ est un sous-espace de dimension $p+1$ de $\mathbb{R}^n$. Supposons à partir de maintenant que les variables $\boldsymbol{x}$ et la variable y sont ici centrées. Notons qu'aucune hypothèse de loi n'est faite dans cette section, les propriétés géométriques découlent des propriétés de l'espérance et de la variance dans l'espace des variables de variance finie.

Avec cette notation, notons que le modèle linéaire s'écrit $m(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{\beta}\rangle$. L'espace $\mathcal{H}_z = \{\boldsymbol{x} \in \mathbb{R}^k : m(\boldsymbol{x}) = z\}$ est un hyperplan (affine) qui sépare l'espace en deux. Définissons l'opérateur de projection orthogonale sur $\mathcal{E}_{\mathcal{X}} = \mathcal{H}_0$, $\Pi_{\mathcal{X}} = \boldsymbol{X}[\boldsymbol{X}^\top \boldsymbol{X}]^{-1}\boldsymbol{X}^\top$. Aussi, la prévision que l'on peut faire pour $\boldsymbol{y}$ est :

$$\widehat{\boldsymbol{y}} = \underbrace{\boldsymbol{X}[\boldsymbol{X}^\top \boldsymbol{X}]^{-1}\boldsymbol{X}^\top}_{\Pi_{\mathcal{X}}}\boldsymbol{y} = \Pi_{\mathcal{X}}\boldsymbol{y}. \tag{9}$$

Comme $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{y} - \widehat{\boldsymbol{y}} = (\mathbb{I} - \Pi_{\mathcal{X}})\boldsymbol{y} = \Pi_{\mathcal{X}^\perp}\boldsymbol{y}$, on note que $\widehat{\boldsymbol{\varepsilon}} \perp \boldsymbol{x}$, que l'on interprétera en disant que les résidus sont un terme d'innovation, imprévisible, au sens où $\Pi_{\mathcal{X}}\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{0}$.

Le théorème de Pythagore s'écrit ici :

$$\|\boldsymbol{y}\|^2 = \|\Pi_{\mathcal{X}}\boldsymbol{y}\|^2 + \|\Pi_{\mathcal{X}^\perp}\boldsymbol{y}\|^2 = \|\Pi_{\mathcal{X}}\boldsymbol{y}\|^2 + \|\boldsymbol{y} - \Pi_{\mathcal{X}}\boldsymbol{y}\|^2 = \|\widehat{\boldsymbol{y}}\|^2 + \|\widehat{\boldsymbol{\varepsilon}}\|^2 \tag{10}$$

qui se traduit classiquement en terme de somme de carrés :

$$\underbrace{\sum_{i=1}^n y_i^2}_{n\times\text{variance totale}} = \underbrace{\sum_{i=1}^n \widehat{y}_i^2}_{n\times\text{variance expliquée}} + \underbrace{\sum_{i=1}^n (y_i - \widehat{y}_i)^2}_{n\times\text{variance résiduelle}}$$

Le coefficient de détermination, R^2 (nous reviendrons sur ce coefficient dans la section 3.2.9) s'interprète alors comme le carré du cosinus de l'angle θ entre $\boldsymbol{y}$ et $\Pi_{\mathcal{X}}\boldsymbol{y}$:

$$R^2 = \frac{\|\Pi_{\mathcal{X}}\boldsymbol{y}\|^2}{\|\boldsymbol{y}\|^2} = 1 - \frac{\|\Pi_{\mathcal{X}^\perp}\boldsymbol{y}\|^2}{\|\boldsymbol{y}\|^2} = \cos^2(\theta).$$

Une application importante a été obtenue par Frisch and Waugh (1933), lorsque l'on partitionne les variables explicatives en deux groupes, $\boldsymbol{X} = [\boldsymbol{X}_1|\boldsymbol{X}_2]$, de telle sorte que la régression devient :

$$\boldsymbol{y} = \beta_0 + \boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{X}_2\boldsymbol{\beta}_2 + \varepsilon$$

Frisch and Waugh (1933) ont montré qu'on pouvait considérer deux projections successives. En effet, si $\boldsymbol{y}_2^\star = \Pi_{\mathcal{X}_1^\perp}\boldsymbol{y}$ et $\boldsymbol{X}_2^\star = \Pi_{\mathcal{X}_1^\perp}\boldsymbol{X}_2$, on peut montrer que

$$\widehat{\boldsymbol{\beta}}_2 = [\boldsymbol{X}_2^{\star\mathsf{T}}\boldsymbol{X}_2^\star]^{-1}\boldsymbol{X}_2^{\star\mathsf{T}}\boldsymbol{y}_2^\star$$

Autrement dit, l'estimation globale est équivalente à l'estimation indépendante des deux modèles si $\boldsymbol{X}_2^\star = \boldsymbol{X}_2$, c'est à dire $\boldsymbol{X}_2 \in \mathcal{E}_{\boldsymbol{X}_1}^\perp$, que l'on peut noter $\boldsymbol{x}_1 \perp \boldsymbol{x}_2$. On obtient ici le théorème de Frisch-Waugh qui garantie que si les variables explicatives entre les deux groupes sont orthogonales, alors l'estimation globale est équivalente à deux régressions indépendantes, sur chacun des jeux de variables explicatives. Ce qui est un théorème de double projection, sur des espaces orthogonaux. Beaucoup de résultats et d'interprétations sont obtenus par des interprétations géométriques (liées fondamentalement aux liens entre l'espérance conditionnelle et la projection orthogonale dans l'espace des variables de variance finie).

Cette vision géométrique permet de mieux comprendre le problème de la sous-identification, c'est à dire le cas où le vrai modèle serait $y_i = \beta_0 + \boldsymbol{x}_1^\mathsf{T}\boldsymbol{\beta}_1 + \boldsymbol{x}_2^\mathsf{T}\boldsymbol{\beta}_2 + \varepsilon_i$, mais le modèle estimé est $y_i = \beta_0 + \boldsymbol{x}_1^\mathsf{T}\boldsymbol{b}_1 + \eta_i$. L'estimateur du maximum de vraisemblance de $\boldsymbol{b}_1$ est :

$$\begin{aligned}
\widehat{\boldsymbol{b}}_1 &= (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{y} \\
&= (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}[\boldsymbol{X}_{1,i}\boldsymbol{\beta}_1 + \boldsymbol{X}_{2,i}\boldsymbol{\beta}_2 + \varepsilon] \\
&= (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1\boldsymbol{\beta}_1 + (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2\boldsymbol{\beta}_2 + (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\varepsilon \\
&= \boldsymbol{\beta}_1 + \underbrace{(\boldsymbol{X}_1'\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2\boldsymbol{\beta}_2}_{\boldsymbol{\beta}_{12}} + \underbrace{(\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\varepsilon}_{\nu_i}
\end{aligned}$$

de telle sorte que $\mathbb{E}[\widehat{\boldsymbol{b}}_1] = \boldsymbol{\beta}_1 + \boldsymbol{\beta}_{12}$, le biais étant nul uniquement dans le cas où $\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2 = \boldsymbol{0}$ (c'est à dire $\boldsymbol{X}_1 \perp \boldsymbol{X}_2$): on retrouve ici une conséquence du théorème de Frisch-Waugh.

En revanche, la sur-identification correspond au cas où le vrai modèle serait $y_i = \beta_0 + \boldsymbol{x}_1^\mathsf{T}\boldsymbol{\beta}_1 + \varepsilon_i$, mais le modèle estimé est $y_i = \beta_0 + \boldsymbol{x}_1^\mathsf{T}\boldsymbol{b}_1 + \boldsymbol{x}_2^\mathsf{T}\boldsymbol{b}_2 + \eta_i$. Dans ce cas, l'estimation est sans biais, au sens où $\mathbb{E}(\widehat{\boldsymbol{b}}_1) = \boldsymbol{\beta}_1$ mais l'estimateur n'est pas efficient. Et comme nous l'avons vu dans la section précédente, il n'est pas rare d'avoir des valeurs de

$\widehat{\boldsymbol{b}}_2$ qui sont considérées comme significativement non-nulles. Nous évoquerons dans la section suivante une méthode efficace de choix de variables (et éviter la sur-identification).

3.2.5 *Du paramétrique au non-paramétrique*

La réécriture de l'équation (9) sous la forme

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}} = \underbrace{\boldsymbol{X}[\boldsymbol{X}^\top\boldsymbol{X}]^{-1}\boldsymbol{X}^\top}_{\Pi_{\mathcal{X}}}\boldsymbol{y}$$

permet de voir la prévision directement comme une transformation linéaire des observations. De manière plus générale, on peut obtenir un prédicteur linéaire en considérant $m(\boldsymbol{x}) = \boldsymbol{s}_{\boldsymbol{x}}^\top\boldsymbol{y}$, où $\boldsymbol{s}_{\boldsymbol{x}}$ est un vecteur de poids, qui dépendent de $\boldsymbol{x}$, interprété comme un vecteur de lissage. En utilisant les vecteurs $\boldsymbol{s}_{\boldsymbol{x}_i}$, calculés à partir des $\boldsymbol{x}_i$, on obtient une matrice $\boldsymbol{S}$ de taille $n \times n$, et $\widehat{\boldsymbol{y}} = \boldsymbol{S}\boldsymbol{y}$. Dans le cas de la régression linéaire décrite auparavant, $\boldsymbol{s}_{\boldsymbol{x}} = \boldsymbol{X}[\boldsymbol{X}^\top\boldsymbol{X}]^{-1}\boldsymbol{x}$, et classiquement, trace($\boldsymbol{S}$) est le nombre de colonnes de la matrice $\boldsymbol{X}$ (le nombre de variables explicatives). Dans ce contexte de prédicteurs linéaires, trace($\boldsymbol{S}$) est souvent vu comme un équivalent au nombre de paramètres (ou complexité, ou dimension, du modèle), et $\nu = n - \text{trace}(\boldsymbol{S})$ est alors le nombre de degrés de liberté (comme défini dans Ruppert and Carroll (2003) et Simonoff (1996)). Le principe de parcimonie[5] consiste à minimiser cette dimension (la trace de la matrice $\boldsymbol{S}$) autant que faire se peut. Mais dans le cas général, cette dimension est plus complexe à définir. Notons que l'estimateur introduit par Nadaraya (1964) et Watson (1964), dans le cas d'une régression non-paramétrique simple, s'écrit également sous cette forme puisque

$$\widehat{m}_h(x) = \boldsymbol{s}_x^\top\boldsymbol{y} = \sum_{i=1}^n s_{x,i}y_i \quad \text{avec} \quad s_{x,i} = \frac{K_h(x-x_i)}{K_h(x-x_1)+\cdots+K_h(x-x_n)},$$

où $K(\cdot)$ est une fonction noyau, qui attribue une valeur d'autant plus faible que x_i est proche de x, et $h > 0$ est la fenêtre de lissage.

L'introduction de ce méta-paramètre h pose un soucis, car il convient de le choisir judicieusement. En faisant des développement limités, on peut montrer que si X a pour densité f,

$$\text{biais}[\widehat{m}_h(x)] = \mathbb{E}[\widehat{m}_h(x)] - m(x) \sim h^2\left(\frac{C_1}{2}m''(x) + C_2 m'(x)\frac{f'(x)}{f(x)}\right)$$

et

$$\text{Var}[\widehat{m}_h(x)] \sim \frac{C_3}{nh}\frac{\sigma(x)}{f(x)}$$

5 « *pluralitas non est ponenda sine necessitate* » pour reprendre le principe énoncé par Guillaume d'Occam (les multiples ne doivent pas être utilisés sans nécessité).

pour des constantes que l'on peut estimer (voir Simonoff (1996) par exemple). Ces deux fonctions évoluent inversement en fonction de h, comme le rappelle la Figure 48 (où le méta-paramètre est ici h^{-1}). L'idée naturelle est alors de chercher à minimiser l'erreur quadratique moyenne, le MSE, $\text{biais}[\widehat{m}_h(x)]^2 + \text{Var}[\widehat{m}_h(x)]$, ce qui donne une valeur optimale pour h de la forme $h^\star = O(n^{-1/5})$, ce qui rappelle la règle de Silverman (1986). En plus grande dimension, pour des variables $\boldsymbol{x}$ continues, on peut utiliser un noyau multivarié, de fenêtre matricielle $\boldsymbol{H}$,

$$\mathbb{E}[\widehat{m}_{\boldsymbol{H}}(\boldsymbol{x})] \sim m(\boldsymbol{x}) + \frac{C_1}{2}\text{trace}\Big(\boldsymbol{H}^\top m''(\boldsymbol{x})\boldsymbol{H}\Big) + C_2\frac{m'(\boldsymbol{x})^\top \boldsymbol{H}\boldsymbol{H}^\top \nabla f(\boldsymbol{x})}{f(\boldsymbol{x})}$$

et

$$\text{Var}[\widehat{m}_{\boldsymbol{H}}(\boldsymbol{x})] \sim \frac{C_3}{n\,\det(\boldsymbol{H})}\frac{\sigma(\boldsymbol{x})}{f(\boldsymbol{x})}.$$

Si $\boldsymbol{H}$ est une matrice diagonale, avec le même terme h sur la diagonale, alors $h^\star = O(n^{-1/(4+\dim(\boldsymbol{x}))})]$. Cela dit, en pratique, on sera davantage intéressé par la version intégrée de l'erreur quadratique,

$$MISE(\widehat{m}_h) = \mathbb{E}[MSE(\widehat{m}_h(X))] = \int MSE(\widehat{m}_h(x))dF(x),$$

dont on peut montrer que

$$MISE[\widehat{m}_h] \sim \overbrace{\frac{h^4}{4}\left(\int x^2k(x)dx\right)^2\int\Big[m''(x)+2m'(x)\frac{f'(x)}{f(x)}\Big]^2dx}^{\text{biais}^2} + \overbrace{\frac{\sigma^2}{nh}\int k^2(x)dx\cdot\int\frac{dx}{f(x)}}^{\text{variance}},$$

lorsque $n\to\infty$ et $nh\to\infty$. On retrouve ici une relation asymptotique qui rappelle l'ordre de grandeur de Silverman (1986),

$$h^\star = n^{-\frac{1}{5}}\left(\frac{C_1\int\frac{dx}{f(x)}}{C_2\int\Big[m''(x)+2m'(x)\frac{f'(x)}{f(x)}\Big]dx}\right)^{\frac{1}{5}},$$

sauf que beaucoup de termes ici sont inconnus. On verra, l'apprentissage automatique propose des techniques computationnelles, lorsque l'économètre avait pris l'habitude de chercher des propriétés asymptotiques.

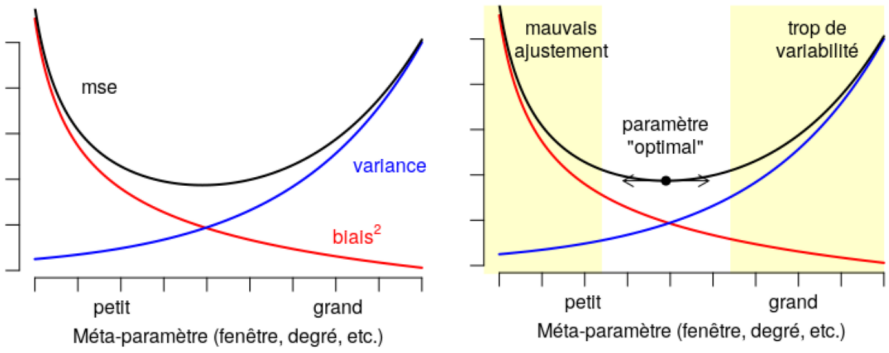


Figure 48.: Choix du méta-paramètre et le problème de Boucle d'Or : il ne doit être ni trop grand (sinon il y a trop de variance), ni trop petit (sinon il y a trop de biais) .

3.2.6 *Famille exponentielle et modèles linéaires*

Le modèle linéaire Gaussien est un cas particulier d'une vaste famille de modèles linéaires, obtenu lorsque la loi conditionnelle de Y appartient à la famille exponentielle

$$f(y_i|\theta_i,\phi) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i,\phi)\right) \quad \text{avec} \quad \theta_i = \psi(\boldsymbol{x}_i^\top\boldsymbol{\beta}).$$

Les fonctions a, b et c sont spécifiées en fonction du type de loi exponentielle (étudiée abondamment en statistique depuis les Darmois (1935), comme le rappelle Brown (1986)), et ψ est une fonction bijective que se donne l'utilisateur. La log-vraisemblance a alors une expression relative simple

$$\log\mathcal{L}(\boldsymbol{\theta},\phi|\boldsymbol{y}) = \prod_{i=1}^{n}\log f(y_i|\theta_i,\phi) = \frac{\sum_{i=1}^n y_i\theta_i - \sum_{i=1}^n b(\theta_i)}{a(\phi)} + \sum_{i=1}^{n} c(y_i,\phi)$$

et la condition du premier ordre s'écrit alors

$$\frac{\partial\log\mathcal{L}(\boldsymbol{\theta},\phi|\boldsymbol{y})}{\partial\boldsymbol{\beta}} = \boldsymbol{X}^\top\boldsymbol{W}^{-1}[\boldsymbol{y}-\widehat{\boldsymbol{y}}] = \boldsymbol{0}$$

pour reprendre les notations de Müller (2011), où $\boldsymbol{W}$ est une matrice de poids (qui dépend de $\boldsymbol{\beta}$). Compte tenu du lien entre θ et l'espérance de Y, au lieu de spécifier la fonction $\psi(\cdot)$, on aura plutôt tendance à spécifier la fonction de lien $g(\cdot)$ définie par

$$\widehat{y} = m(\boldsymbol{x}) = \mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] = g^{-1}(\boldsymbol{x}^\top\boldsymbol{\beta}).$$

Pour la régression linéaire Gaussienne on prendra un lien Identité, alors que pour la régression de Poisson, le lien naturel (dit canonique) est le lien logarithmique. Ici, comme $\boldsymbol{W}$ dépend de $\boldsymbol{\beta}$ (avec $\boldsymbol{W} = \text{diag}(\nabla g(\widehat{\boldsymbol{y}})\text{Var}[\boldsymbol{y}])$) il n'existe en général par de formule explicite pour l'estimateur du maximum de vraisemblance. Mais un algorithme itératif permet d'obtenir une approximation numérique. En posant

$$\boldsymbol{z} = g(\widehat{\boldsymbol{y}}) + (\boldsymbol{y} - \widehat{\boldsymbol{y}}) \cdot \nabla g(\widehat{\boldsymbol{y}})$$

correspondant au terme d'erreur d'un développement de Taylor à l'ordre 1 de g, on obtient un algorithme de la forme

$$\widehat{\boldsymbol{\beta}}_{k+1} = [\boldsymbol{X}^\top \boldsymbol{W}_k^{-1} \boldsymbol{X}]^{-1} \boldsymbol{X}^\top \boldsymbol{W}_k^{-1} \boldsymbol{z}_k$$

En itérant, on notera $\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}_\infty$, et on peut montrer que - moyennant quelques hypothèses techniques (cf Müller (2011)) - cet estimateur est asymptotiquement Gaussien, avec

$$\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{\mathcal{L}} \mathcal{N}(\boldsymbol{0}, I(\boldsymbol{\beta})^{-1}),$$

où numériquement $I(\boldsymbol{\beta}) = \phi \cdot [\boldsymbol{X}^\top \boldsymbol{W}_\infty^{-1} \boldsymbol{X}]$.

D'un point de vue numérique toujours, on résout la condition du premier ordre, et la loi de Y n'intervient pas réellement. Par exemple, on peut estimer une « régression de Poisson » même lorsque $y \in \mathbb{R}_+$, pas nécessairement $y \in \mathbb{N}$. Autrement dit, la loi de Y n'est qu'une interprétation donnée ici, et l'algorithme pourrait être introduit de manière différente (comme nous le verrons dans la section suivante), sans forcément avoir de modèle probabiliste sous-jacent.

3.2.7 *Régression logistique*

La régression logistique est le modèle linéaire généralisé obtenu avec une loi de Bernoulli, et une fonction de lien qui est la fonction quantile d'une loi logistique (ce qui correspond au lien canonique au sens de la famille exponentielle). Compte tenu de la forme de la loi de Bernoulli, l'économétrie propose un modèle pour $y_i \in \{0, 1\}$, dans lequel le logarithme de la cote suit un modèle linéaire :

$$\log\left(\frac{\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}]}{\mathbb{P}[Y \neq 1|\boldsymbol{X} = \boldsymbol{x}]}\right) = \beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta},$$

ou encore :

$$\mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}] = \mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \frac{e^{\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}}}{1 + e^{\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}}} = H(\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}), \quad \text{où} \quad H(\cdot) = \frac{\exp(\cdot)}{1 + \exp(\cdot)},$$

correspondant à la fonction de répartition de la loi logistique. L'estimation de $(\beta_0, \boldsymbol{\beta})$ se fait par maximisation de la vraisemblance :

$$\mathcal{L} = \prod_{i=1}^{n} \left(\frac{e^{\boldsymbol{x}_i^\mathsf{T}\beta}}{1+e^{\boldsymbol{x}_i^\mathsf{T}\beta}} \right)^{y_i} \left(\frac{1}{1+e^{\boldsymbol{x}_i^\mathsf{T}\beta}} \right)^{1-y_i}$$

On continuera à parler des modèles linéaires car les courbes d'isoprobabilités sont ici les hyperplans parallèles $b_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}$. à ce modèle, popularisé par Berkson (1944), certains préféreront le modèle probit (comme le raconte Berkson (1951)), introduit par Bliss (1934). Dans ce modèle :

$$\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] = \mathbb{P}[Y=1|\boldsymbol{X}=\boldsymbol{x}] = \Phi(\beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}),$$

où Φ désigne la fonction de répartition de la loi normale centrée réduite. Ce modèle présente l'avantage d'avoir un lien direct avec le modèle linéaire Gaussien, puisque

$$y_i = \mathbf{1}(y_i^\star > 0) \quad \text{avec} \quad y_i^\star = \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} + \varepsilon_i$$

où les résidus sont Gaussiens, de loi $\mathcal{N}(0,\sigma^2)$. Un alternative est d'avoir des résidus centrés de variance unitaire, et de considérer une modélisation latente de la forme $y_i = \mathbf{1}(y_i^\star > \xi)$ (où ξ sera à fixer). On le voit, ces techniques sont fondamentalement liées à un modèle stochastique sous-jacent. Mais dans la section 3.5, nous présenterons plusieurs techniques alternatives - tirées de la littérature en apprentissage - pour ce problème de classification (avec deux classes, ici 0 et 1).

3.2.8 *Régression en grande dimension*

Comme nous l'avions mentionné auparavant, la condition du premier ordre $\boldsymbol{X}^\mathsf{T}(\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}) = \mathbf{0}$ se résout numériquement en effectuant une décomposition QR, pour un coût en $O(np^2)$ opérations (où p est le rang de $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$). Numériquement, ce calcul peut être long (soit parce que p est grand, soit parce que n est grand), et une stratégie plus simple peut être de faire du sous-échantillonnage. Soit $n_s \ll n$, et considérons un sous-échantillon de taille n_s de $\{1,\cdots,n\}$. Alors $\widehat{\boldsymbol{\beta}}_s = (\boldsymbol{X}_s^\mathsf{T}\boldsymbol{X}_s)^{-1}\boldsymbol{X}_s^\mathsf{T}\boldsymbol{y}_s$ est une bonne approximation de $\widehat{\boldsymbol{\beta}}$ comme le montre Dhillon et al. (2014). Cet algorithme est toutefois dangereux si certains points ont un pouvoir de levier important (i.e. $L_i = \boldsymbol{x}_i(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{x}_i^\mathsf{T}$). Tobin (2011) propose de transformer les données (de manière linéaire), mais une approche plus populaire est de faire du sous-échantillonnage non uniforme, avec une probabilité liée à l'influence des observations (définie par $I_i = \widehat{\varepsilon}_i L_i/(1-L_i)^2$, et qui malheureusement ne peut être calculée qu'une fois le modèle estimé).

De manière générale, on parlera de données massives lorsque la table de données de taille $n \times p$ ne tient pas en mémoire RAM de l'ordinateur. Cette situation est souvent rencontrée en apprentissage statistique de nos jours avec très souvent $p \ll n$. C'est la

raison pour laquelle, en pratique de nombreuses bibliothèques d'algorithmes assimilées à de l'apprentissage machine[6] utilisent des méthodes itératives pour résoudre la condition du premier ordre. Lorsque le modèle paramétrique à calibrer est effectivement convexe et semi-différentiable, il est possible d'utiliser par exemple la méthode de descente de gradient stochastique comme le suggère Bottou (2010). Ce dernier permet de s'affranchir à chaque itération du calcul du gradient sur chaque observation de notre base d'apprentissage. Plutôt que d'effectuer une descente moyenne à chaque itération, on commence par tirer (sans remise) une observation X_i parmi les n disponibles. On corrige ensuite les paramètres du modèle de sorte à ce que la prédiction faite à partir de X_i soit la plus proche possible de la vraie valeur y_i. On réitère ensuite la méthode jusqu'à avoir parcourue l'ensemble des données. Dans cet algorithme il y a donc autant d'itération que d'observations. Contrairement à l'algorithme de descente de gradient (ou méthode de Newton) à chaque itération un seul vecteur de gradient est calculé (et non plus n). Il est néanmoins parfois nécessaire d'exécuter cette algorithme plusieurs fois pour augmenter la convergence des paramètres du modèle.

Si l'objectif est par exemple de minimiser l'erreur quadratique ℓ entre l'estimateur $f_\beta(X)$ et y l'algorithme peut se résumer ainsi :
Étape 0: Mélange des données
Étape d'itérations: Pour $t = 1, ..., n$, on tire $i \in \{1, \cdots, n\}$ sans remise, on pose

$$\beta^{t+1} = \beta^t - \gamma_t \frac{\partial l(y_i, f_{\beta^t}(X_i))}{\partial \beta}$$

Cet algorithme peut être réitéré plusieurs fois dans son ensemble selon le besoin de l'utilisateur. L'avantage de cette méthode est qu'à chaque itération, il n'est pas nécessaire de calculer le gradient sur toutes les observations (plus de somme). Elle est donc adaptée aux bases de données volumineuses. Cet algorithme s'appuie sur une convergence en probabilité vers un voisinage de l'optimum (et non pas l'optimum lui même).

3.2.9 *Qualité d'un ajustement et choix de modèle*

Dans le modèle linéaire Gaussien, le coefficient de détermination - noté R^2 - est souvent utilisé comme mesure de la qualité d'ajustement. Compte tenu de la formule de décomposition de la variance

$$\underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}_{\text{variance totale}} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}_{\text{variance résiduelle}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - \bar{y})^2}_{\text{variance expliquée}}$$

6 comme, par exemple, celles du langage Python.

on définit le R^2 comme le ratio de variance expliquée et de la variance totale, autre interprétation du coefficient que nous avions introduit à partir de la géométrie des moindres carrés.

$$R^2 = \frac{\sum_{i=1}^n (y_i - \bar{y})^2 - \sum_{i=1}^n (y_i - \widehat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Les sommes des carrés d'erreurs dans cette écriture peut se réécrire comme une log-vraisemblance. Or rappelons qu'à une constante près, dans les modèles linéaires généralisés, la déviance est définie par

$$\text{Deviance}(\boldsymbol{\beta}) = -2\log[\mathcal{L}]$$

que l'on peut aussi noter $\text{Deviance}(\widehat{\boldsymbol{y}})$. On peut définir une déviance nulle comme celle obtenue sans utiliser les variables explicatives $\boldsymbol{x}$, de telle sorte que $\widehat{y}_i = \overline{y}$. On peut alors définir, dans un contexte plus général

$$R^2 = \frac{\text{Deviance}(\overline{y}) - \text{Deviance}(\widehat{\boldsymbol{y}})}{\text{Deviance}(\overline{y})} = 1 - \frac{\text{Deviance}(\widehat{\boldsymbol{y}})}{\text{Deviance}(\overline{y})}.$$

Toutefois, cette mesure ne peut être utilisée pour choisir un modèle, si on souhaite avoir au final un modèle relativement simple, car elle augmente artificiellement avec l'ajout de variables explicatives sans effet significatif. On aura alors tendance à préférer le R^2 ajusté

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p} = R^2 - \underbrace{(1 - R^2)\frac{p-1}{n-p}}_{\text{pénalisation}},$$

où p est le nombre de paramètres du modèle (noté plus généralement ν dans la section 3.2.5). à la mesure de la qualité de l'ajustement, on va pénaliser les modèles trop complexes.

Cette idée va se retrouver dans le critère d'Akaike, où $AIC = \text{Deviance} + 2 \cdot p$ ou dans le critère de Schwarz, $BIC = \text{Deviance} + \log(n) \cdot p$. En grande dimension (typiquement $p > \sqrt{n}$), on aura tendance à utiliser un AIC corrigé, défini par

$$AICc = \text{Deviance} + 2 \cdot p \cdot \frac{n}{n-p-1}$$

Ces critères sont utilisés dans les méthodes dites « *stepwise* », introduisant les méthodes ensemblistes. Dans la méthode dite « *forward* », on commence par régresser sur la constante, puis on ajoute une variable à la fois, en retenant celle qui fait le plus baisser le critère AIC, jusqu'à ce que rajouter une variable augmente le critère AIC du modèle. Dans la méthode dite « *backward* », on commence par régresser sur toutes les variables, puis on enlève une variable à la fois, en retirant celle qui fait le plus baisser le critère AIC, jusqu'à ce que retirer une variable augmente le critère AIC du modèle.

Une autre justification de cette notion de pénalisation (nous reviendrons sur cette idée en apprentissage) peut être la suivante. Considérons un estimateur dans la classe des prédicteurs linéaires,

$$\mathcal{M} = \Big\{ m : \ m(\boldsymbol{x}) = s_h(\boldsymbol{x})^\mathsf{T}\boldsymbol{y} \text{ où } S = (s(\boldsymbol{x}_1), \cdots, s(\boldsymbol{x}_n)^\mathsf{T} \text{ est la matrice de lissage} \Big\}$$

et supposons que $\boldsymbol{y} = m_0(\boldsymbol{x}) + \boldsymbol{\varepsilon}$, avec $\mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{0}$ et $\text{Var}[\boldsymbol{\varepsilon}] = \sigma^2\mathbb{I}$, de telle sorte que $m_0(\boldsymbol{x}) = \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$. D'un point de vue théorique, le risque quadratique, associé à un modèle estimé $\widehat{m}$, s'écrit

$$\begin{aligned}
\mathcal{R}(\widehat{m}) &= \mathbb{E}\Big[(Y - \widehat{m}(\boldsymbol{X}))^2\Big] \\
&= \underbrace{\mathbb{E}\Big[(Y - m_0(\boldsymbol{X}))^2\Big]}_{\text{erreur}} + \underbrace{\mathbb{E}\Big[(m_0(\boldsymbol{X}) - \mathbb{E}[\widehat{m}(\boldsymbol{X})])^2\Big]}_{\text{biais}} + \underbrace{\mathbb{E}\Big[(\mathbb{E}[\widehat{m}(\boldsymbol{X})] - \widehat{m}(\boldsymbol{X}))^2\Big]}_{\text{variance}}
\end{aligned}$$

si m_0 désigne le vrai modèle. Le premier terme est parfois appelé « erreur de Bayes », et ne dépend pas de l'estimateur retenu, $\widehat{m}$.

Le risque empirique quadratique, associé à un modèle m, est ici :

$$\widehat{\mathcal{R}}_n(m) = \frac{1}{n}\sum_{i=1}^n (y_i - m(\boldsymbol{x}_i))^2 = \frac{1}{n}\|\boldsymbol{y} - m(\boldsymbol{x})\|^2$$

(par convention). On reconnaît ici l'erreur quadratique moyenne, MSE, qui donnera plus généralement le « risque » du modèle m quand on utilise une autre fonction de perte (comme nous le discuterons dans la partie suivante). Notons que:

$$\mathbb{E}[\widehat{\mathcal{R}}_n(m)] = \frac{1}{n}\|m_0(\boldsymbol{x}) - m(\boldsymbol{x})\|^2 + \frac{1}{n}\mathbb{E}\Big(\|\boldsymbol{y} - m_0(\boldsymbol{x})\|^2\Big)$$

On peut montrer que :

$$n\mathbb{E}\Big[\widehat{\mathcal{R}}_n(\widehat{m})\Big] = \mathbb{E}\Big(\|\boldsymbol{y} - \widehat{m}(\boldsymbol{x})\|^2\Big) = \|(\mathbb{I} - \boldsymbol{S})m_0\|^2 + \sigma^2\|\mathbb{I} - \boldsymbol{S}\|^2$$

de telle sorte que le (vrai) risque de $\widehat{m}$ est :

$$\mathcal{R}_n(\widehat{m}) = \mathbb{E}\Big[\widehat{\mathcal{R}}_n(\widehat{m})\Big] + 2\frac{\sigma^2}{n}\text{trace}(\boldsymbol{S}).$$

Aussi, si $\text{trace}(\boldsymbol{S}) \geq 0$, le risque empirique sous-estime le vrai risque de l'estimateur. On reconnaît ici le nombre de degrés de liberté du modèle, le terme de droite correspondant au C_p de Mallow, introduit dans Mallows (1973) utilisant non pas la déviance mais le R^2.

3.2.10 *Econométrie et tests statistiques*

Le test le plus classique en économétrie est probablement le test de significativité, correspondant à la nullité d'un coefficient dans un modèle de régression linéaire. Formellement, il s'agit du test de $H_0 : \beta_k = 0$ contre $H_1 : \beta_k \neq 0$. Le test de Student, basé sur la statistique $t_k = \widehat{\beta}_k / \text{se}_{\widehat{\beta}_k}$, permet a priori de trancher entre les deux alternatives, à l'aide de la p-value du test, définie par $\mathbb{P}[|T| > |t_k|]$ avec $T \overset{\mathcal{L}}{\sim} Std_\nu$, où ν est le nombre de degrés de liberté du modèle ($\nu = p + 1$ pour le modèle linéaire standard). En grande dimension, cette statistique est néanmoins d'un intérêt très limité, compte tenu d'un FDR (« *False Discovery Ratio* ») important. Classiquement, avec un niveau de significativité $\alpha = 0.05$, 5% des variables sont faussement significatives. Supposons que nous disposions de $p = 100$ variables explicatives, mais que 5 (seulement) sont réellement significatives. On peut espérer que ces 5 variables passeront le test de Student, mais on peut aussi s'attendre à ce que 5 variables supplémentaires (test faussement positif) ressortent. On aura alors 10 variables perçues comme significatives, alors que seulement la moitié le sont, soit un ratio FDR de 50%. Afin d'éviter cet écueil récurent dans les tests multiples, il est naturel d'utiliser la procédure de Benjamini and Hochberg (1995).

3.2.11 *Quitter la corrélation pour quantifier un effet causal*

Les modèles économétriques sont utilisés pour mettre en oeuvre des politiques publiques. Il est alors fondamental de bien comprendre les mécanismes sous-jacents pour savoir quelles variables permettent effectivement d'agir sur une variable d'intérêt. Mais on passe alors dans une autre dimension importante de l'économétrie. C'est à Jerry Neyman que l'on doit les premiers travaux sur l'identification de mécanismes causaux, c'est Rubin (1974) qui a formalisé le test, appelé « modèle causal de Rubin » dans Holland (1986). Les premières approches autour de la notion de causalité en économétrie se sont faites avec l'utilisation des variables instrumentales, des modèles avec discontinuité de régression, l'analyse de différences dans les différences, ainsi que des expériences naturelles ou pas. La causalité est généralement déduite en comparant l'effet d'une politique - ou plus généralement d'un traitement - avec son contre-factuel, idéalement donné par un groupe témoin, aléatoire. L'effet causal du traitement est alors défini comme $\Delta = y_1 - y_0$, c'est à dire la différence entre ce que serait la situation avec traitement (noté $t = 1$) et sans traitement (noté $t = 0$). Le souci est que seul $y = t \cdot y_1 + (1 - t)y_0$ et t sont observés. Autrement dit l'effet causal de la variable t sur y n'est pas observé (puisque seule une des deux variables potentielles - y_0 ou y_1 est observée pour chaque individu), mais il est aussi individuel, et donc fonction de covariables $\boldsymbol{x}$. Généralement, en faisant des hypothèses sur la distribution du triplet (Y_0, Y_1, T), certains paramètres de la distribution de l'effet causal deviennent identifiables, à partir de la densité des variables observables (Y, T). Classiquement, on sera intéressé par les moments de cette distribution, en particulier l'effet moyen du traitement dans la population, $\mathbb{E}[\Delta]$, voire juste l'effet moyen du

traitement en cas de traitement $\mathbb{E}[\Delta|T=1]$. Si le résultat (Y_0, Y_1) est indépendant de la variable d'accès au traitement T, on peut montrer que $\mathbb{E}[\Delta] = \mathbb{E}[Y|T=1] - \mathbb{E}[Y|T=0]$. Mais si cette hypothèse d'indépendance n'est pas vérifiée, on a un biais de sélection, souvent associé à $\mathbb{E}[Y_0|T=1] - \mathbb{E}[Y_0|T=0]$. Rosenbaum and Rubin (1983) proposent d'utiliser un score de propension à être traité, $p(\boldsymbol{x}) = \mathbb{P}[T=1|\boldsymbol{X}=x]$, en notant que si la variable Y_0 est indépendante de l'accès au traitement T conditionnellement aux variables explicatives $\boldsymbol{X}$, alors elle est indépendante de T conditionnellement au score $p(\boldsymbol{X})$: il suffit de les apparier à l'aide de leur score de propension. Heckman and Vytlacil (2003) propose ainsi un estimateur à noyau sur le score de propension, ce qui permet d'avoir simplement un estimateur de l'effet du traitement, conditionnellement au fait d'être traité.

3.3 PHILOSOPHIE DES MÉTHODES D'APPRENTISSAGE AUTOMATIQUE

Parallèlement à ces outils développés par et pour des économistes, toute une littérature a été développée sur des questions similaires, centrées autour de la prévision. Pour Breiman (2001a), une première différence vient du fait que la statistique s'est développée autour du principe d'inférence (ou d'expliciter la relation liant y aux variables $\boldsymbol{x}$) alors qu'une autre culture s'intéresse avant tout à la prédiction. Dans une discussion qui suit l'article, David Cox l'affirme très clairement « *predictive success* (...) *is not the primary basis for model choice* ». Nous allons présenter les fondements des techniques d'apprentissage automatique (les exemples d'algorithmes étant présentés dans les sections suivantes). Le point important, comme nous allons le voir, est que la principale préoccupation de l'apprentissage machine est liée aux propriétés de généralisation d'un modèle, c'est-à-dire sa performance - selon un critère choisi a priori - sur des données nouvelles, et donc des tests hors échantillon.

3.3.1 *Apprentissage par une machine*

Aujourd'hui, on parle d' « apprentissage automatique » pour décrire tout un ensemble de techniques, souvent computationnelles, alternatives à l'approche décrite auparavant (correspondant à l'économétrie classique). Avant de les caractériser autant que possible, notons juste qu'historiquement d'autres noms ont pu être donnés. Par exemple, Friedman (1997) propose de faire le lien entre la statistique (qui ressemble beaucoup aux techniques économétriques - test d'hypothèses, ANOVA, régression linéaire, logistique, GLM, etc) et ce qu'il appelait alors « *data mining* » (qui englobait alors les arbres de décisions, les méthodes des plus proches voisins, les réseaux de neurones, etc.). Le pont qu'il contribuera à construire correspond aux techniques d'apprentissages statistiques, décrites dans Hastie and Friedman (2009), mais l'apprentissage machine est un très vaste champ de recherche.

L'apprentissage dit « naturel » (par opposition à celui d'une machine) est celui des enfants, qui apprennent à parler, à lire, à jouer. Apprendre à parler signifie segmenter et catégoriser des sons, et les associer à des significations. Un enfant apprend aussi simultanément la structure de sa langue maternelle et acquiert un ensemble de mots décrivant le monde qui l'entoure. Plusieurs techniques sont possibles, allant d'un apprentissage par coeur, par généralisation, par découverte, apprentissage plus ou moins supervisé ou autonome, etc. L'idée en intelligence artificielle est de s'inspirer du fonctionnement du cerveau pour apprendre, pour permettre un apprentissage « artificiel » ou « automatique », par une machine. Une première application a été d'apprendre à une machine à jouer à un jeux (*tic-tac-toe*, échecs, go, etc). Une étape indispensable est d'expliquer l'objectif qu'il doit atteindre pour gagner. Une approche historique a été de lui apprendre les règles du jeu. Si cela permet de jouer, cela ne permettra pas à la machine de *bien* jouer. En supposant que la machine connaisse les règles du jeu, et qu'elle a le choix entre plusieurs dizaines de coups possible, lequel doit-elle choisir ? L'approche classique en intelligence artificielle utilise l'algorithme dit *min-max* utilisant une fonction d'évaluation : dans cet algorithme, la machine effectue une recherche en avant dans l'arbre des coups possibles, aussi loin que les ressources de calcul le lui permettent (une dizaine de coups aux échecs, par exemple). Ensuite, elle calcule différents critères (qui lui ont été indiqués au préalable) pour toutes les positions (nombre de pièces prises, ou perdues, occupation du centre, etc. dans notre exemple du jeu d'échec), et finalement, la machine joue le coup qui lui permet de maximiser son gain. Un autre exemple peut être celui de la classification et de la reconnaissance d'images ou de formes. Par exemple, la machine doit identifier un chiffre dans une écriture manuscrite (chèque, code postal). Il s'agit de prédire la valeur d'une variable y, en sachant qu'a priori $y \in \{0, 1, 2, \cdots, 8, 9\}$. Un stratégie classique est de fournir à la machine des bases d'apprentissage, autrement dit ici des millions d'images labélisées (identifiées) de chiffres manuscrits. Une stratégie simple et naturelle est d'utiliser un critère de décision basé sur les plus proches voisins dont on connaît l'étiquette (à l'aide d'une métrique prédéfinie).

La méthode des plus proches voisins (« *k-nearest neighbors* ») peut être décrit de la manière suivante : on considère (comme dans la partie précédente) un ensemble de n observations, c'est à dire des paires $(y_i, \boldsymbol{x}_i)$ avec $\boldsymbol{x}_i \in \mathbb{R}^p$. Considérons une distance Δ sur $\mathbb{R}^p$ (la distance Euclidienne ou la distance de Mahalanobis, par exemple). étant donnée une nouvelle observation $\boldsymbol{x} \in \mathbb{R}^p$, supposons les observations ordonnées en fonction de la distance entre les $\boldsymbol{x}_i$ et $\boldsymbol{x}$, au sens où

$$\Delta(\boldsymbol{x}_1, \boldsymbol{x}) \leq \Delta(\boldsymbol{x}_2, \boldsymbol{x}) \leq \cdots \leq \Delta(\boldsymbol{x}_n, \boldsymbol{x})$$

alors on peut considérer comme prédiction pour y la moyenne des k plus proches voisins,

$$m_k(\boldsymbol{x}) = \frac{1}{k} \sum_{i=1}^{k} y_i.$$

L'apprentissage fonctionne ici par induction, à partir d'un échantillon (appelé base d'apprentissage).

L'apprentissage automatique englobe ces algorithmes qui donnent aux ordinateurs la capacité d'apprendre sans être explicitement programmé (comme l'avait défini Arthur Samuel en 1959). La machine va alors explorer les données avec un objectif précis (comme chercher les plus proches voisins dans l'exemple que nous venons de décrire). Tom Mitchell a proposé une définition plus précise en 1998 : on dit qu'un programme d'ordinateur apprend de l'expérience E par rapport à une tâche T et une mesure de performance P, si sa performance sur T, mesurée par P, s'améliore avec l'expérience E. La tâche T peut être un score de défaut par exemple, et la performance P peut être le pourcentage d'erreurs commise. Le système apprend si le pourcentage de défauts prédit augmente avec l'expérience.

On le voit, l'apprentissage machine est fondamentalement un problème d'optimisation d'un critère à partir de données (dites d'apprentissage). Nombreux sont les ouvrages de programmation qui proposent des algorithmes, sans jamais faire mention d'un quelconque modèle probabiliste. Dans Watt and Katsaggelos (2016) par exemple, il n'est fait mention du mot « probabilité » qu'une seule fois, avec cette note de bas de page qui surprendra et fera sourire les économètres, « *logistic regression can also be interpreted from a probabilistic perspective* » (page 86). Mais beaucoup d'ouvrages récents proposent une relecture des approches d'apprentissage machine à l'aide de théories probabilistes, suite aux travaux de Vaillant et Vapnik. En proposant le paradigme de l'apprentissage « *probablement à peu près correct* » (PAC), une saveur probabiliste a été rajouté à l'approche jusqu'alors très computationnelle, en quantifiant l'erreur de l'algorithme d'apprentissage (dans un problème de classification).

3.3.2 *Le tournant des années 80/90 et le formalisme probabiliste*

On dispose d'un échantillon d'apprentissage, avec des observations $(\boldsymbol{x}_i, y_i)$ où les variables y sont dans un ensemble $\mathcal{Y}$. Dans le cas de la classification, $\mathcal{Y} = \{-1, +1\}$, mais on peut imaginer un ensemble relativement général. Un prédicteur est une fonction m à valeurs dans $\mathcal{Y}$, permettant d'étiqueter (ou de classer) les nouvelles observations à venir. On suppose que les étiquettes sont produites par un classifieur f appelé cible. Pour un statisticien, cette fonction serait le vrai modèle. Naturellement, on veut construire m le plus proche possible de f. Soit $\mathbb{P}$ une distribution (inconnue) sur $\mathcal{X}$. L'erreur de m relativement à la cible f est définie par

$$\mathcal{R}_{\mathbb{P},f}(m) = \mathbb{P}[m(\boldsymbol{X}) \neq f(\boldsymbol{X})] \text{ où } \boldsymbol{X} \sim \mathbb{P},$$

ou écrit de manière équivalente,

$$\mathcal{R}_{\mathbb{P},f}(m) = \mathbb{P}\Big[\{\boldsymbol{x} \in \mathcal{X} : m(\boldsymbol{x}) \neq f(\boldsymbol{x})\}\Big].$$

Pour trouver notre classifieur, il devient nécessaire de supposer qu'il existe un lien entre les données de notre échantillon et le couple $(\mathbb{P}, f)$, c'est à dire un modèle de génération des données. On va alors supposer que les $\boldsymbol{x}_i$ sont obtenus par des tirages indépendants suivant $\mathbb{P}$, et qu'ensuite $y_i = f(\boldsymbol{x}_i)$.

On peut ici définir le risque empirique d'un modèle m,

$$\widehat{\mathcal{R}}(m) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(m(\boldsymbol{x}_i) \neq y_i).$$

Il est alors important d'admettre qu'on ne peut pas trouver un modèle parfait, au sens où $\mathcal{R}_{\mathbb{P},f}(m) = 0$. En effet, si on considère le cas le plus simple qui soit, avec $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ et que $\mathbb{P}$ soit telle que $\mathbb{P}(\{\boldsymbol{x}_1\}) = p$ et $\mathbb{P}(\{\boldsymbol{x}_2\}) = 1 - p$. Il est aussi possible d'observer x_1 et x_2, et malgré tout, de se tromper sur les étiquettes. Aussi, au lieu de chercher un modèle parfait, on peut tenter d'avoir un modèle approximativement correct. On va alors chercher à trouver m tel que $\mathcal{R}_{\mathbb{P},f}(m) \leq \epsilon$, où ϵ est un seuil spécifié a priori.

Une fois admis ce premier écueil, qui fait penser à l'erreur de modèle, notons aussi un second problème. Sur notre exemple à deux valeurs, la probabilité de ne jamais observer x_2 parmi n tirages suivant $\mathbb{P}$ est p^n. Il sera alors impossible de trouver $m(x_2)$ car cette valeur n'aura jamais été observée. Autrement dit, aucun algorithme ne peut nous assurer d'avoir avec certitude, avec n observations, $\mathcal{R}_{\mathbb{P},f}(m) \leq \epsilon$. On va alors chercher à être probablement approximativement correct (PAC). Pour se faire, on autorise l'algorithme à se tromper avec une probabilité δ, là aussi fixée a priori.

Aussi, quand on construit un classifieur, on ne connaît ni $\mathbb{P}$, ni f, mais on se donne un critère de précision ϵ, et un paramètre de confiance δ, et on dispose de n observations. Notons que n, ϵ et δ peuvent être liés. On cherche alors un modèle m tel que $\mathcal{R}_{\mathbb{P},f}(m) \leq \epsilon$ avec probabilité (au moins) $1 - \delta$, de manière à être probablement approximativement correct.

Wolpert (1997) a montré (détaillé dans Wolpert (1997)) qu'il n'existe pas d'algorithme d'apprentissage universel. En particulier, on peut montrer qu'il existe $\mathbb{P}$ telle que $\mathcal{R}_{\mathbb{P},f}(m)$ soit relativement grande, avec une probabilité (elle aussi) relativement grande.

L'interprétation qui en est faite est qu'il est nécessaire d'avoir un biais pour apprendre. Comme on ne peut pas apprendre (au sens PAC) sur l'ensemble des fonctions m, on va alors contraindre m à appartenir une classe particulière, notée $\mathcal{M}$. Supposons pour commencer que $\mathcal{M}$ contienne un nombre fini de modèles possibles. On peut alors montrer que pour tout ϵ et δ, que pour tout $\mathbb{P}$ et f, si on dispose d'assez d'observations (plus précisément $n \geq \epsilon^{-1}\log[\delta^{-1}|\mathcal{M}|]$, alors avec une probabilité plus grande que $1 - \delta$, $\mathcal{R}_{\mathbb{P},f}(m^\star) \leq \epsilon$ où

$$m^\star \in \underset{m\in\mathcal{M}}{\operatorname{argmin}}\left\{\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(m(\boldsymbol{x}_i) \neq y_i)\right\}$$

autrement dit $m^\star$ est un modèle dans $\mathcal{M}$ qui minimise le risque empirique.

Un peut aller un peu plus loin, en restant dans le cas où $\mathcal{Y}=\{-1,+1\}$. Une classe $\mathcal{M}$ de classifieurs sera dite PAC-apprenable s'il existe $n_{\mathcal{M}}:[0,1]^2\to\mathbb{N}$ tel que, pour tout ϵ, δ, $\mathbb{P}$ et si on suppose que la cible f appartient à $\mathcal{M}$, alors en utilisant $n>n_{\mathcal{M}}(\epsilon,\delta)$ tirages d'observations $\boldsymbol{x}_i$ suivant $\mathbb{P}$, étiquetés y_i par f, alors il existe $m\in\mathcal{M}$ tel que, avec probabilité $1-\delta$, $\mathcal{R}_{\mathbb{P},f}(m)\leq\epsilon$. La fonction $n_{\mathcal{M}}$ est alors appelée complexité d'échantillon pour apprendre. En particulier, nous avons vu que si $\mathcal{M}$ contient un nombre fini de classifieurs, alors $\mathcal{M}$ est PAC-apprenable avec la complexité $n_{\mathcal{M}}(\epsilon,\delta)=\epsilon^{-1}\log[\delta^{-1}|\mathcal{M}|]$.

Naturellement, on souhaiterait avoir un résultat plus général, en particulier si $\mathcal{M}$ n'est pas fini. Pour cela, il faut utiliser la dimension VC de Vapnik-Chervonenkis, qui repose sur l'idée de pulvérisation de nuages de points (pour une classification binaire). Considérons k points $\{\boldsymbol{x}_1,\cdots\boldsymbol{x}_k\}$, et considérons l'ensemble

$$\mathcal{E}_k=\Big\{(m(\boldsymbol{x}_1),\cdots,m(\boldsymbol{x}_k))\text{ pour }m\in\mathcal{M})\Big\}.$$

Notons que les éléments de $\mathcal{E}_k$ appartiennent à $\{-1,+1\}^k$. Autrement dit $|\mathcal{E}_k|\leq 2^k$. On dira que $\mathcal{M}$ pulvérise l'ensemble des points si toutes les combinaisons sont possibles, c'est à dire $|\mathcal{E}_k|=2^k$. Intuitivement, les étiquettes de l'ensemble de points ne procurent pas assez d'information sur la cible f, car tout est possible. La dimension VC de $\mathcal{M}$ est alors

$$VC(\mathcal{M})=\sup\Big\{k\text{ tel que }\mathcal{M}\text{ pulvérise }\{\boldsymbol{x}_1,\cdots\boldsymbol{x}_k\}\Big\}.$$

Par exemple si $\mathcal{X}=\mathbb{R}$ et que l'on considère l'ensemble des modèles (simples) de la forme $m_{a,b}=\mathbf{1}_{\pm}(x\in[a,b])$. Aucun ensemble de points $\{x_1,x_2,x_3\}$ ordonnés ne peut être pulvérisé car il suffit d'assigner respectivement $+1$, -1 et $+1$ à x_1, x_2 et x_3 respectivement, donc $VC<3$. En revanche $\{0,1\}$ est pulvérisé donc $VC\geq 2$. La dimension de cet ensemble de prédicteur est 2. Si on augmente d'une dimension, $\mathcal{X}=\mathbb{R}^2$ et que l'on considère l'ensemble des modèles (simples) de la forme $m_{\boldsymbol{a},\boldsymbol{b}}=\mathbf{1}_{\pm}(x\in[\boldsymbol{a},\boldsymbol{b}])$ (où $[\boldsymbol{a},\boldsymbol{b}]$ désigne le rectangle), alors la dimension de $\mathcal{M}$ est ici 4.

Pour introduire les SVM, plaçons nous dans le cas où $\mathcal{X}=\mathbb{R}^k$, et considérons des séparations par des hyperplans passant par l'origine (on dira homogènes), au sens où $m_{\boldsymbol{w}}(\boldsymbol{x})=\mathbf{1}_{\pm}(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}\geq 0)$. On peut montrer qu'aucun ensemble de $k+1$ points ne peut être pulvérisé par ces deux espaces homogènes dans $\mathbb{R}^k$, et donc $VC(\mathcal{M})=k$. Si on rajoute une constante, au sens où $m_{\boldsymbol{w},b}(\boldsymbol{x})=\mathbf{1}_{\pm}(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}+b\geq 0)$, on peut montrer qu'aucun ensemble de $k+2$ points ne peut être pulvérisé par ces deux espaces (non homogènes) dans $\mathbb{R}^k$, et donc $VC(\mathcal{M})=k+1$.

De cette dimension VC, on déduit le théorème dit fondamental de l'apprentisssage : si $\mathcal{M}$ est une classe de dimension $d=VC(\mathcal{M})$, alors il existe des constante positives $\underline{C}$ et $\overline{C}$ telles que la complexité d'échantillon pour que $\mathcal{M}$ soit PAC-apprenable vérifie

$$\underline{C}\epsilon^{-1}\Big(d+\log[\delta^{-1}]\Big)\leq n_{\mathcal{M}}(\epsilon,\delta)\leq\overline{C}\epsilon^{-1}\Big(d\log[\epsilon^{-1}]+\log[\delta^{-1}]\Big).$$

Le lien entre la notion d'apprentissage (tel que défini dans Vailiant (1984)) et la dimension VC a été clairement établi dans Blumer and Warmuth (1989).

Néanmoins, si les travaux de Vapnik et Chervonenkis sont considérés comme fondateurs de l'apprentissage statistique, il convient de citer aussi les travaux de Thomas Cover dans les années 60 et 70, en particulier Cover (1965) sur les capacités des modèles linéaires et Cover and Hart (1967) sur l'apprentissage dans le contexte de l'algorithme des k plus proches voisins. Ces travaux ont fait le lien entre l'apprentissage, la théorie de l'information (avec l'ouvrage de référence Cover and Thomas (1991)), la complexité et la statistique. D'autres auteurs ont rapprochés les deux communautés, de l'apprentissage et de la statistique par la suite. Par exemple Halbert White proposait de voir les réseaux de neurones dans un contexte statistique dans White (1989), allant jusqu'à affirmer que « *learning procedures used to train artificial neural networks are inherently statistical techniques. It follows that statistical theory can provide considerable insight into the properties, advantages, and disadvantages of different network learning methods* ». Ce tournant à la fin des années 80 ancrera la théorie de l'apprentissage dans un contexte probabiliste.

3.3.3 *Le choix de l'objectif et la fonction de perte*

Ces choix (de l'objectif et de la fonction de perte) sont essentiels, et très dépendants du problème considéré. Commençons par décrire un modèle historiquement important, le « perceptron » de Rosenblatt (1960), introduit dans des problèmes de classification, où $y \in \{-1, +1\}$, inspiré par McCullogh and Pitts (1943). On dispose de données $\{(y_i, \boldsymbol{x}_i)\}$, et on va construire de manière itérative un ensemble de modèles $m_k(\cdot)$, où à chaque étape, on va apprendre des erreurs du modèle précédent. Dans le perceptron, on considère un modèle linéaire de telle sorte que :

$$m(\boldsymbol{x}) = \mathbf{1}_{\pm}(\beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} \geq 0) = \left\{ \begin{array}{l} +1 \text{ si } \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} \geq 0 \\ -1 \text{ si } \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} < 0 \end{array} \right. ,$$

où les coefficients $\boldsymbol{\beta}$ sont souvent interprétés comme des « poids » attribués à chacune des variables explicatives. On se donne des poids initiaux $(\beta_0^{(0)}, \boldsymbol{\beta}^{(0)})$, que l'on va mettre à jour en tenant compte de l'erreur de prédiction commise, entre y_i et la prédiction $\widehat{y}_i^{(k)}$:

$$\widehat{y}_i^{(k)} = m^{(k)}(\boldsymbol{x}_i) = \mathbf{1}_{\pm}(\beta_0^{(k)} + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}^{(k)} \geq 0),$$

avec, dans le cas du perceptron :

$$\beta_j^{(k+1)} = \beta_j^{(k)} + \eta \underbrace{(\boldsymbol{y} - \widehat{\boldsymbol{y}}^{(k)})^\mathsf{T}}_{=\ell(\boldsymbol{y}, \widehat{\boldsymbol{y}}^{(k)})} \boldsymbol{x}_j$$

où ici $\ell(y, y') = \mathbf{1}(y \neq y')$ est une fonction de perte, qui permettra de donner un prix à une erreur commise, en prédisant $y' = m(\boldsymbol{x})$ et en observant y. Pour un problème de régression, on peut considérer une erreur quadratique ℓ_2, telle que $\ell(y, m(\boldsymbol{x})) = (y - m(\boldsymbol{x}))^2$ ou en valeur absolue ℓ_1, avec $\ell(y, m(\boldsymbol{x})) = |y - m(\boldsymbol{x})|$. Ici, pour notre problème de classification, nous utilisions une indicatrice de mauvaise qualification (on pourrait discuter le caractère symétrique de cette fonction de perte, laissant croire qu'un faux positif coûte autant qu'un faux négatif). Une fois spécifiée cette fonction de perte, on reconnaît dans le problème décrit auparavant une descente de gradient, et on voit que l'on cherche à résoudre :

$$m^\star(\boldsymbol{x}) = \underset{m\in\mathcal{M}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, m(\boldsymbol{x}_i)) \right\} \tag{11}$$

pour un ensemble de prédicteurs $\mathcal{M}$ prédéfini. Tout problème d'apprentissage machine est mathématiquement formulé comme un problème d'optimisation, dont la solution détermine un ensemble de paramètres de modèle (si la famille $\mathcal{M}$ est décrite par un ensemble de paramètres - qui peuvent être des coordonnées dans une base fonctionnelle). On pourra noter $\mathcal{M}_0$ l'espace des hyperplans de $\mathbb{R}^p$ au sens où

$$m \in \mathcal{M}_0 \quad \text{signifie} \quad m(\boldsymbol{x}) = \beta_0 + \boldsymbol{\beta}^\mathsf{T}\boldsymbol{x} \quad \text{avec} \quad \boldsymbol{\beta} \in \mathbb{R}^p,$$

engendrant la classe des prédicteurs linéaires. On aura alors l'estimateur qui minimise le risque empirique. Une partie des travaux récents en apprentissage statistique vise à étudier les propriétés de l'estimateur $\widehat{m}^\star$, dit « oracle », dans une famille d'estimateurs $\mathcal{M}$,

$$\widehat{m}^\star = \underset{\widehat{m}\in\mathcal{M}}{\operatorname{argmin}} \Big\{ \mathcal{R}(\widehat{m}, m) \Big\}.$$

Cet estimateur est, bien entendu, impossible à définir car il dépend de m, le vrai modèle, inconnu.

Mais revenons un peu davantage sur ces fonctions de perte. Une fonction de perte ℓ est une fonction $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, symmétrique, qui vérifie l'inégalité triangulaire, et telle que $\ell(\boldsymbol{x}, \boldsymbol{y}) = 0$ si et seulement si $\boldsymbol{x} = \boldsymbol{y}$. La norme associée est $\|\cdot\|$, telle que $\ell(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| = \ell(\boldsymbol{x} - \boldsymbol{y}, \mathbf{0})$ (en utilisant le fait que $\ell(\boldsymbol{x}, \boldsymbol{y} + \boldsymbol{z}) = \ell(\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{z})$ - nous reverrons cette propriété fondamentale par la suite).

Pour une fonction de perte quadratique, on notera que l'on peut avoir une interprétation particulière de ce problème, puisque :

$$\overline{y} = \underset{m\in\mathbb{R}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \frac{1}{n}[y_i - m]^2 \right\} = \underset{m\in\mathbb{R}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell_2(y_i, m) \right\},$$

où ℓ_2 est la distance quadratique usuelle Si l'on suppose - comme on le faisait en économétrie - qu'il existe un modèle probabiliste sous-jacent, et en notant que :

$$\mathbb{E}(Y) = \underset{m\in\mathbb{R}}{\text{argmin}} \left\{ \|Y-m\|_{\ell_2}^2 \right\} = \underset{m\in\mathbb{R}}{\text{argmin}} \left\{ \mathbb{E}\left([Y-m]^2 \right) \right\} = \underset{m\in\mathbb{R}}{\text{argmin}} \left\{ \mathbb{E}\Big[\ell_2(Y,m) \Big] \right\}$$

on notera que ce que l'on essaye d'obtenir ici, en résolvant le problème (11) en prenant pour ℓ la norme ℓ_2, est une approximation (dans un espace fonctionnel donné, $\mathcal{M}$) de l'espérance conditionnelle $\boldsymbol{x} \mapsto \mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}]$. Une autre fonction de perte particulièrement intéressante est la perte ℓ_1, $\ell_1(y,m) = |y-m|$. Rappelons que

$$\text{médiane}(\boldsymbol{y}) = \underset{m\in\mathbb{R}}{\text{argmin}} \left\{ \sum_{i=1}^n \ell_1(y_i,m) \right\}.$$

Le problème d'optimisation :

$$\widehat{m}^\star = \underset{m\in\mathcal{M}_0}{\text{argmin}} \left\{ \sum_{i=1}^n |y_i - m(\boldsymbol{x}_i)| \right\}$$

est obtenu en économétrie si on suppose que la loi conditionnelle de Y suit une loi de Laplace centrée sur $m(\boldsymbol{x})$, et en maximisant la (log) vraisemblance (la somme des valeurs absolues des erreurs correspond à la log-vraisenblance d'une loi de Laplace). On pourra noter d'ailleurs que si la loi conditionnelle de Y est symétrique par rapport à 0, la médiane et la moyenne coïncident Si on réécrit cette fonction de perte $\ell_1(y,m) = |(y-m)(1/2 - \mathbf{1}_{y\leq m})|$, on peut obtenir une généralisation pour $\tau \in (0,1)$:

$$\widehat{m}_\tau^\star = \underset{m\in\mathcal{M}_0}{\text{argmin}} \left\{ \sum_{i=1}^n \ell_\tau^{\mathsf{q}}(y_i, m(\boldsymbol{x}_i)) \right\} \quad \text{avec} \quad \ell_\tau^{\mathsf{q}}(x,y) = (x-y)(\tau - \mathbf{1}_{x\leq y})$$

est alors la régression quantile de niveau τ (voir Koenker (2003) et d'Haultefœuille and Givord (2014)). Une autre fonction de perte, introduite par Aigner and Schmidt (1977) et analysée dans Waltrup et al. (2014), est la fonction associée à la notion d'expectiles :

$$\ell_\tau^{\mathsf{e}}(x,y) = (x-y)^2 \cdot \Big| \tau - \mathbf{1}_{x\leq y} \Big|$$

avec $\tau \in [0,1]$. On voit le parallèle avec la fonction quantile :

$$\ell_\tau^{\mathsf{q}}(x,y) = |x-y| \cdot \Big| \tau - \mathbf{1}_{x\leq y} \Big|.$$

Koenker and Machado (1999) et Yu and Moyeed (2001) ont d'ailleurs noté un lien entre cette condition et la recherche du maximum de vraisemblance lorsque la loi conditionnelle de Y suit une loi de Laplace assymétrique.

En lien avec cette approche, Gneiting (2011) a introduit la notion de « *statistique ellicitable* » - ou de « *mesure ellicitable* » dans sa version probabiliste (ou distributionnelle) : T sera dite « *ellicitable* » s'il existe une fonction de perte $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ telle que :

$$T(Y) = \underset{x \in \mathbb{R}}{\operatorname{argmin}} \left\{ \int_{\mathbb{R}} \ell(x, y) dF(y) \right\} = \underset{x \in \mathbb{R}}{\operatorname{argmin}} \left\{ \mathbb{E}\Big[\ell(x, Y)\Big] \text{ où } Y \overset{\mathcal{L}}{\sim} F \right\}.$$

La moyenne (espérance mathématique) est ainsi ellicitable par la distance quadratique, ℓ_2, alors que la médiane est ellicitable par la distance ℓ_1. Selon Gneiting (2011), cette propriété est essentielle pour construire des prédictions. Il peut alors exister un lien fort entre des mesures associées à des modèles probabilistes et les fonctions de perte. Enfin, la statistique Bayésienne propose un lien direct entre la forme de la loi a priori et la fonction de perte, comme l'ont étudié Berger (1985) et Bernardo and Smith (2000). Nous reviendrons sur l'utilisation de ces différentes normes dans la section sur la pénalisation.

3.3.4 *Boosting et apprentissage séquentiel*

Nous l'avons vu auparavant: la modélisation repose ici sur la résolution d'un problème d'optimisation, et résoudre le problème décrit par l'équation (11) est d'autant plus complexe que l'espace fonctionnel $\mathcal{M}$ est volumineux. L'idée du Boosting, tel qu'introduit par Shapire and Freund (2012), est d'apprendre, lentement, à partir des erreurs du modèle, de manière itérative. à la première étape, on estime un modèle m_1 pour $\boldsymbol{y}$, à partir de $\boldsymbol{X}$, qui donnera une erreur $\boldsymbol{\varepsilon}_1$. à la seconde étape, on estime un modèle m_2 pour $\boldsymbol{\varepsilon}_1$, à partir de $\boldsymbol{X}$, qui donnera une erreur $\boldsymbol{\varepsilon}_2$, etc. On va alors retenir comme modèle, au bout de k itération :

$$m^{(k)}(\cdot) = \underbrace{m_1(\cdot)}_{\sim \boldsymbol{y}} + \underbrace{m_2(\cdot)}_{\sim \boldsymbol{\varepsilon}_1} + \underbrace{m_3(\cdot)}_{\sim \boldsymbol{\varepsilon}_2} + \cdots + \underbrace{m_k(\cdot)}_{\sim \boldsymbol{\varepsilon}_{k-1}} = m^{(k-1)}(\cdot) + m_k(\cdot). \tag{12}$$

Ici, l'erreur ε est vue comme la différence entre y et le modèle $m(\boldsymbol{x})$, mais elle peut aussi être vue comme le gradient associé à la fonction de perte quadratique. Formellement, $\boldsymbol{\varepsilon}$ peut être vu comme un $\nabla \ell$ dans un contexte plus général (on retrouve ici une interprétation qui fait penser aux résidus dans les modèles linéaires généralisés).

L'équation (12) peut se voir comme une descente du gradient, mais écrit de manière duale. En effet, la descente de gradient permettant d'obtenir le minimum d'une fonction f repose sur une équation de la forme

$$\underbrace{f(\boldsymbol{x}_k)}_{\langle f, \boldsymbol{x}_k \rangle} \sim \underbrace{f(\boldsymbol{x}_{k-1})}_{\langle f, \boldsymbol{x}_{k-1} \rangle} + \underbrace{(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})}_{\alpha_k} \underbrace{\nabla f(\boldsymbol{x}_{k-1})}_{\langle \nabla f, \boldsymbol{x}_{k-1} \rangle}$$

Le problème (11) est dual dans le sens où c'est la fonction f qui doit être optimisée. On pourrait alors écrire une descente de gradient de la forme :

$$\underbrace{f_k(\boldsymbol{x})}_{\langle f_k,\boldsymbol{x}\rangle} \sim \underbrace{f_{k-1}(\boldsymbol{x})}_{\langle f_{k-1},\boldsymbol{x}\rangle} + \underbrace{(f_k - f_{k-1})}_{\beta_k} \underbrace{\star}_{\langle f_{k-1},\nabla\boldsymbol{x}\rangle}$$

où le terme $\star$ peut être interprété comme un gradient, mais dans un espace fonctionnel, et non plus dans $\mathbb{R}^p$. Le problème (12) va alors se réécrire comme un problème d'optimisation :

$$m^{(k)} = m^{(k-1)} + \underset{h\in\mathcal{H}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell(\underbrace{y_i - m^{(k-1)}(\boldsymbol{x}_i)}_{\varepsilon_{k,i}}, h(\boldsymbol{x}_i)) \right\} \tag{13}$$

où l'astuce consiste à considérer un espace $\mathcal{H}$ relativement simple (on parlera de « *weak learner* »). Classiquement, les fonctions $\mathcal{H}$ sont des fonctions en escalier (que l'on retrouvera dans les arbres de classification et de régression) appelées « *stumps* ». Afin de s'assurer que l'apprentissage est effectivement lent, il n'est pas rare d'utiliser un paramètre de « shrinkage », et au lieu de poser, par exemple, $\varepsilon_1 = y - m_1(\boldsymbol{x})$, on posera $\varepsilon_1 = y - \alpha \cdot m_1(\boldsymbol{x})$ avec $\alpha \in [0, 1]$. On notera que c'est parce qu'on utilise pour $\mathcal{H}$ un espace non-linéaire, et que l'apprentissage est lent, que cet algorithme fonctionne bien. Dans le cas du modèle linéaire Gaussien, rappelons en effet que les résidus $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ sont orthogonaux aux variables explicatives, $\boldsymbol{X}$, et il est alors impossible d'apprendre de nos erreurs. La principale difficulté est de s'arrêter à temps, car après trop d'itérations, ce n'est plus la fonction m que l'on approxime, mais le bruit. Ce problème est appelé sur-apprentissage.

Cette présentation a l'avantage d'avoir une heuristique faisant penser à un modèle économétrique, en modélisant de manière itérative les résidus par un modèle (très) simple. Mais ce n'est souvent pas la présentation retenue dans la littérature en apprentissage, qui insiste davantage sur une heuristique d'algorithme d'optimisation (et d'approximation du gradient). La fonction est apprise de manière itérative, en partant d'une valeur constante,

$$m^{(0)} = \underset{m\in\mathbb{R}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell(y_i, m) \right\}.$$

puis on considère l'apprentissage suivant

$$m^{(k)} = m^{(k-1)} + \underset{h\in\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(y_i, m^{(k-1)}(\boldsymbol{x}_i) + h(\boldsymbol{x}_i)), \tag{14}$$

qui peut s'écrire, si $\mathcal{H}$ est un ensemble de fonctions différentiables,

$$m^{(k)} = m^{(k-1)} - \gamma_k \sum_{i=1}^{n} \nabla_{m^{(k-1)}} \ell(y_i, m^{(k-1)}(\boldsymbol{x}_i)), \tag{15}$$

où

$$\gamma_k = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(y_i, m^{(k-1)}(\boldsymbol{x}_i) - \gamma \nabla_{m^{(k-1)}} \ell(y_i, m^{(k-1)}(\boldsymbol{x}_i))\right).$$

Pour mieux comprendre le lien avec l'approche décrite auparavant, à l'étape k, on définit des pseudo-résidus en posant

$$r_{i,k} = -\left.\frac{\partial \ell(y_i, m(\boldsymbol{x}_i))}{\partial m(\boldsymbol{x}_i)}\right|_{m(\boldsymbol{x})=m^{(k-1)}(\boldsymbol{x})} \text{ pour } i = 1, \cdots, n.$$

On cherche alors un modèle simple pour expliquer ces pseudo-résidus en fonction des variables explicatives $\boldsymbol{x}_i$, i.e. $r_{i,k} = h^\star(\boldsymbol{x}_i)$, où $h^\star \in \mathcal{H}$. Dans un second temps, on cherche un multiplicateur optimal en résolvant

$$\gamma_k = \underset{\gamma \in \mathbb{R}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell(y_i, m^{(k-1)}(\boldsymbol{x}_i) + \gamma h^\star(\boldsymbol{x}_i)) \right\}$$

puis on met à jour le modèle en posant $m_k(\boldsymbol{x}) = m_{k-1}(\boldsymbol{x}) + \gamma_k h^\star(\boldsymbol{x})$. Plus formellement, on passe de l'équation (13) - qui montre clairement qu'on construit un modèle sur les résidus - à l'équation (14) - qui sera ensuite retraduit comme une problème de calcul de gradient - en notant que $\ell(y, m + h) = \ell(y - m, h)$. Classiquement, les fonctions $\mathcal{H}$ sont construites avec des arbres de régression. Il est aussi possible d'utiliser une forme de pénalisation en posant $m_k(\boldsymbol{x}) = m_{k-1}(\boldsymbol{x}) + \nu\gamma_k h^\star(\boldsymbol{x})$, avec $\nu \in (0, 1)$. Mais revenons un peu plus longuement sur l'importance de la pénalisation avant de discuter les aspects numériques de l'optimisation.

3.3.5 *Pénalisation et choix de variables*

Dans la section 3.2.9, nous avions évoqué le principe de parcimonie, populaire en économétrie. Le critère d'Akaike était basé sur une pénalisation de la vraisemblance en tenant compte de la complexité du modèle (le nombre de variables explicatives retenues). Si en économétrie, il est d'usage de maximiser la vraisemblance (pour construire un estimateur asymptotiquement sans biais), et de juger de la qualité du modèle ex-post en pénalisant la vraisemblance, la stratégie ici sera de pénaliser ex-ante dans la fonction objectif, quitte à construire un estimateur biaisé. Typiquement, on va construire :

$$(\widehat{\beta}_{0,\lambda}, \widehat{\boldsymbol{\beta}}_\lambda) = \operatorname{argmin} \left\{ \sum_{i=1}^{n} \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) + \lambda \text{ pénalisation}(\boldsymbol{\beta}) \right\}, \tag{16}$$

où la fonction de pénalisation sera souvent une norme $\|\cdot\|$ choisie a priori, et un paramètre de pénalisation λ (on retrouve en quelque sorte la distinction entre AIC et BIC si la fonction de pénalisation est la complexité du modèle - le nombre de variables explicatives retenues). Dans le cas de la norme ℓ_2, on retrouve l'estimateur RIDGE, et pour la norme ℓ_1, on retrouve l'estimateur LASSO (« *Least Absolute Shrinkage and Selection Operator* »). La pénalisation utilisée auparavant faisait intervenir le nombre de degrés de liberté du modèle, il peut alors paraître surprenant de faire intervenir $\|\boldsymbol{\beta}\|_{\ell_2}$ comme dans la régression RIDGE. On peut toutefois envisager une vision Bayésienne de cette pénalisation. Rappelons que dans un modèle Bayésien :

$$\underbrace{\mathbb{P}[\boldsymbol{\theta}|\boldsymbol{y}]}_{\text{a posteriori}} \propto \underbrace{\mathbb{P}[\boldsymbol{y}|\boldsymbol{\theta}]}_{\text{vraisemblance}} \cdot \underbrace{\mathbb{P}[\boldsymbol{\theta}]}_{\text{a priori}} \quad \text{soit} \quad \log \mathbb{P}[\boldsymbol{\theta}|\boldsymbol{y}] = \underbrace{\log \mathbb{P}[\boldsymbol{y}|\boldsymbol{\theta}]}_{\text{log vraisemblance}} + \underbrace{\log \mathbb{P}[\boldsymbol{\theta}]}_{\text{pénalisation}}.$$

Dans un modèle linéaire Gaussien, si on suppose que la loi *a priori* de $\boldsymbol{\theta}$ suit une loi normale centrée, on retrouve une pénalisation basée sur une forme quadratique des composantes de $\boldsymbol{\theta}$.

Avant de revenir en détails sur ces deux estimateurs, obtenus en utilisant la norme ℓ_1 ou la norme ℓ_2, revenons un instant sur un problème très proche : celui du meilleur choix de variables explicatives. Classiquement (et ça sera encore plus vrai en grande dimension), on peut disposer d'un grand nombre de variables explicatives, p, mais beaucoup sont juste du bruit, au sens où $\beta_j = 0$ pour un grand nombre de j. Soit s le nombre de covariables (réellement) pertinentes, $s = \#\mathcal{S}$ avec $\mathcal{S}(\boldsymbol{\beta}) = \{j = 1, \cdots, p; \beta_j \neq 0\}$. Si on note $\boldsymbol{X}_{\mathcal{S}}$ la matrice constituée des variables pertinentes (en colonnes), alors on suppose que le vrai modèle est de la forme $y = \boldsymbol{x}_{\mathcal{S}}^{\mathsf{T}}\boldsymbol{\beta}_{\mathcal{S}} + \varepsilon$. Intuitivement, un estimateur intéressant serait alors $\widehat{\boldsymbol{\beta}}_{\mathcal{S}} = [\boldsymbol{X}_{\mathcal{S}}^{\mathsf{T}}\boldsymbol{X}_{\mathcal{S}}]^{-1}\boldsymbol{X}_{\mathcal{S}}\boldsymbol{y}$, mais cet estimateur n'est que théorique car $\mathcal{S}$ est ici inconnue. Cet estimateur est l'estimateur oracle évoqué auparavant. On peut alors être tenté de résoudre

$$(\widehat{\beta}_{0,s}, \widehat{\boldsymbol{\beta}}_s) = \text{argmin}\left\{\sum_{i=1}^{n} \ell(y_i, \beta_0 + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta})\right\}, \text{ sous la contrainte } \#\mathcal{S}(\boldsymbol{\beta}) = s.$$

Ce problème a été introduit par Foster and George (1994) en introduisant la norme ℓ_0. Plus précisément, définissons ici les trois normes suivantes

$$\|\boldsymbol{a}\|_{\ell_0} = \sum_{i=1}^{d} \mathbf{1}(a_i \neq 0), \quad \|\boldsymbol{a}\|_{\ell_1} = \sum_{i=1}^{d} |a_i| \quad \text{et} \quad \|\boldsymbol{a}\|_{\ell_2} = \left(\sum_{i=1}^{d} a_i^2\right)^{1/2}, \text{ pour } \boldsymbol{a} \in \mathbb{R}^d.$$

Considérons les problèmes d'optimisation de la Table 12. Si on considère le problème classique où ℓ est la norme quadratique, les deux problèmes de l'équation ($\ell 1$) de la Table 12 sont équivalents, au sens où, pour toute solution $(\boldsymbol{\beta}^\star, s^\star)$ au problème de gauche, il existe $\lambda^\star$ tel que $(\boldsymbol{\beta}^\star, \lambda^\star)$ soit solution du problème de droite; et inversement. Le résultat

	optimisation contrainte	pénalisation	
meilleur groupe	$\underset{\boldsymbol{\beta};\|\boldsymbol{\beta}\|_{\ell_0}\leq s}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) \right\}$	$\underset{\boldsymbol{\beta},\lambda}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_{\ell_0} \right\}$	$(\ell 0)$
Lasso	$\underset{\boldsymbol{\beta};\|\boldsymbol{\beta}\|_{\ell_1}\leq s}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) \right\}$	$\underset{\boldsymbol{\beta},\lambda}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_{\ell_1} \right\}$	$(\ell 1)$
Ridge	$\underset{\boldsymbol{\beta};\|\boldsymbol{\beta}\|_{\ell_2}\leq s}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) \right\}$	$\underset{\boldsymbol{\beta},\lambda}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \ell(y_i, \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_{\ell_2} \right\}$	$(\ell 2)$

Table 12.: Optimisation contrainte et régularisation.

est également vrai pour les problèmes $(\ell 2)$[7]. Il s'agit en effet de problèmes convexes. En revanche, les deux problèmes $(\ell 0)$ ne sont pas équivalents : si pour $(\boldsymbol{\beta}^\star, \lambda^\star)$ solution du problème de droite, il existe $s^\star$ tel que $\boldsymbol{\beta}^\star$ soit solution du problème de gauche, la réciproque n'est pas vraie. Plus généralement, si on veut utiliser une norme ℓ_p, la sparsité est obtenue si $p \leq 1$ alors qu'il faut avoir $p \geq 1$ pour avoir la convexité du programme d'optimisation.

On peut être tenté de résoudre le programme pénalisé $(\ell 0)$ directement, comme le suggère Foster and George (1994). Numériquement, c'est un problème combinatoire complexe en grande dimension (Natarajan (1995) note que c'est un problème NP-difficile), mais il est possible de montrer que si $\lambda \sim \sigma^2 \log(p)$, alors

$$\mathbb{E}\Big([\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}} - \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big) \leq \underbrace{\mathbb{E}\Big(\boldsymbol{x}_S^\mathsf{T}\widehat{\boldsymbol{\beta}}_S - \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big)}_{=\sigma^2 \# S} \cdot \Big(4\log p + 2 + o(1)\Big).$$

Notons que dans ce cas

$$\widehat{\boldsymbol{\beta}}_{\lambda,j}^{\mathsf{sub}} = \begin{cases} 0 \text{ si } j \notin \mathcal{S}_\lambda(\boldsymbol{\beta}) \\ \widehat{\boldsymbol{\beta}}_j^{\mathsf{ols}} \text{ si } j \in \mathcal{S}_\lambda(\boldsymbol{\beta}), \end{cases}$$

où $\mathcal{S}_\lambda(\boldsymbol{\beta})$ désigne l'ensemble des coordonnées non nulle lors de la résolution de $(\ell 0)$.

Le problème $(\ell 2)$ est strictement convexe si ℓ est la norme quadratique, autrement dit, l'estimateur Ridge est toujours bien défini, avec en plus une forme explicite pour l'estimateur,

$$\widehat{\boldsymbol{\beta}}_\lambda^{\mathsf{ridge}} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I})^{-1}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})\widehat{\boldsymbol{\beta}}^{\mathsf{ols}}.$$

Aussi, on peut en déduire que

$$\text{biais}[\widehat{\boldsymbol{\beta}}_\lambda^{\mathsf{ridge}}] = -\lambda[\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I}]^{-1}\,\widehat{\boldsymbol{\beta}}^{\mathsf{ols}} \text{ et } \text{Var}[\widehat{\boldsymbol{\beta}}_\lambda^{\mathsf{ridge}}] = \sigma^2[\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I}]^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{X}[\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I}]^{-1}.$$

7 Pour $(\ell 1)$, s'il y a équivalence au niveau théorique, il peut exister des soucis numériques car il n'y a pas forcément unicité de la solution.

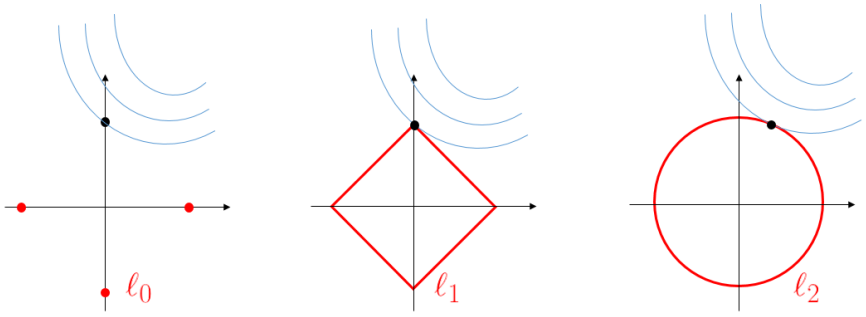


Figure 49.: Pénalisation basée sur la norme ℓ_0, ℓ_1 et ℓ_2 de $\boldsymbol{\beta}$, respectivement (inspiré de Hastie *et al.* (2016)).

Avec une matrice de variables explicatives orthonormées (i.e. $\boldsymbol{X}^\mathsf{T}\boldsymbol{X} = \mathbb{I}$), les expressions se simplifient

$$\text{biais}[\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}}] = \frac{\lambda}{1+\lambda}\,\widehat{\boldsymbol{\beta}}^{\text{ols}} \text{ et } \text{Var}[\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}}] = \frac{\sigma^2}{(1+\lambda)^2}\mathbb{I}.$$

Notons que $\text{Var}[\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}}] < \text{Var}[\widehat{\boldsymbol{\beta}}^{\text{ols}}]$. En notant que

$$\text{mse}[\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}}] = \frac{k\sigma^2}{(1+\lambda)^2} + \frac{\lambda^2}{(1+\lambda)^2}\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta},$$

on obtient une valeur optimale pour λ: $\lambda^\star = k\sigma^2/\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}$.

En revanche, si ℓ n'est plus la norme quadratique mais la norme ℓ_1, le problème $(\ell 1)$ n'est pas toujours strictement convexe, et en particulier, l'optimum n'est pas toujours unique (par exemple si $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ est singulière). Mais le fait que ℓ soit strictement convexe $\boldsymbol{X}\widehat{\boldsymbol{\beta}}$ sera unique. Notons de plus que deux solutions sont forcément cohérentes en terme de signe des coefficients : il n'est pas possible d'avoir $\widehat{\beta}_j < 0$ pour une solution et $\widehat{\beta}_j > 0$ pour une autre. D'un point de vue heuristique, le programme $(\ell 1)$ est intéressant car il permet d'obtenir dans bon nombre de cas une solution en coin, qui correspond à une résolution de problème de type $(\ell 0)$ - comme le montre de manière visuelle la Figure 49.

Considérons un modèle très simple: $y_i = x_i\beta + \varepsilon$, avec une pénalité ℓ_1 et une fonction de perte ℓ_2. Le problème $(\ell 2)$ sécrit alors

$$\min\left\{\boldsymbol{y}^\mathsf{T}\boldsymbol{y} - 2\boldsymbol{y}^\mathsf{T}\boldsymbol{x}\beta + \beta\boldsymbol{x}^\mathsf{T}\boldsymbol{x}\beta + 2\lambda|\beta|\right\}$$

La condition du premier ordre est alors

$$-2\boldsymbol{y}^\mathsf{T}\boldsymbol{x} + 2\boldsymbol{x}^\mathsf{T}\boldsymbol{x}\widehat{\beta} \pm 2\lambda = 0.$$

le signe du dernier terme dépend du signe de $\widehat{\beta}$. Supposons que l'estimateur par moindre carrés (obtenu en posant $\lambda = 0$) soit (strictement positif), autrement dit $\boldsymbol{y}^\mathsf{T}\boldsymbol{x} > 0$. Si λ n'est pas trop grand, on peut imaginer que $\widehat{\beta}$ soit du même signe que $\widehat{\beta}^{\mathsf{mco}}$, et donc la condition devient

$$-2\boldsymbol{y}^\mathsf{T}\boldsymbol{x} + 2\boldsymbol{x}^\mathsf{T}\boldsymbol{x}\widehat{\beta} + 2\lambda = 0.$$

et la solution est

$$\widehat{\beta}_\lambda^{\mathsf{lasso}} = \frac{\boldsymbol{y}^\mathsf{T}\boldsymbol{x} - \lambda}{\boldsymbol{x}^\mathsf{T}\boldsymbol{x}}.$$

En augmentant λ, on va arriver à un moment où $\widehat{\beta}_\lambda = 0$. Si on augmente encore un peu $\widehat{\beta}_\lambda$ ne devient pas négatif car dans ce cas le dernier terme de la condition du premier ordre change, et dans ce cas on cherche à résoudre

$$-2\boldsymbol{y}^\mathsf{T}\boldsymbol{x} + 2\boldsymbol{x}^\mathsf{T}\boldsymbol{x}\widehat{\beta} - 2\lambda = 0.$$

dont la solution est alors

$$\widehat{\boldsymbol{\beta}}_\lambda^{\mathsf{lasso}} = \frac{\boldsymbol{y}^\mathsf{T}\boldsymbol{x} + \lambda}{\boldsymbol{x}^\mathsf{T}\boldsymbol{x}}.$$

Mais cette solution est positive (nous avions supposé $\boldsymbol{y}^\mathsf{T}\boldsymbol{x} > 0$), et donc il est possible d'avoir en même temps $\widehat{\beta}_\lambda < 0$. Aussi, au bout d'un moment, $\widehat{\beta}_\lambda = 0$, qui est alors une solution de coin. Les choses sont bien entendu plus compliquées en dimension plus grande (Tibshirani and Wasserman (2016) revient longuement sur la géométrie des solutions) mais comme le note Candès and Plan (2009), sous des hypothèses minimales garantissant que les prédicteurs ne sont pas fortement corrélés, le Lasso obtient une erreur quadratique presque aussi bonne que si l'on dispose d'un oracle fournissant des informations parfaites sur l"ensembles des β_j qui sont non nuls. Moyennant quelques hypothèses techniques supplémentaires, on peut montrer que cet estimateur est « sparsistant » au sens où le support de $\widehat{\boldsymbol{\beta}}_\lambda^{\mathsf{lasso}}$ est celui de $\boldsymbol{\beta}$, autrement dit Lasso a permis de faire de la sélection de variables (plus de discussions sur ce point peuvent être obtenues dans Hastie and Tibshiriani (2016)).

De manière plus générale, on peut montrer que $\widehat{\boldsymbol{\beta}}_\lambda^{\mathsf{lasso}}$ est un estimateur biaisé, mais qui peut être de variance suffisamment faible pour que l'erreur quadratique moyenne soit plus faible qu'en utilisant des moindres carrés. Pour comparer les trois techniques, par rapport à l'estimateur par moindre carrées (obtenu quand $\lambda = 0$), si on suppose que les variables explicatives sont orthonormées, alors

$$\widehat{\beta}_{\lambda,j}^{\mathsf{sub}} = \widehat{\beta}_j^{\mathsf{ols}}\mathbf{1}_{|\widehat{\beta}_{\lambda,j}^{\mathsf{sub}}|>b}, \quad \widehat{\beta}_{\lambda,j}^{\mathsf{ridge}} = \frac{\widehat{\beta}_j^{\mathsf{ols}}}{1+\lambda} \quad \text{et} \quad \widehat{\beta}_{\lambda,j}^{\mathsf{lasso}} = \text{signe}[\widehat{\beta}_j^{\mathsf{ols}}] \cdot (|\widehat{\beta}_j^{\mathsf{ols}}| - \lambda)_+.$$

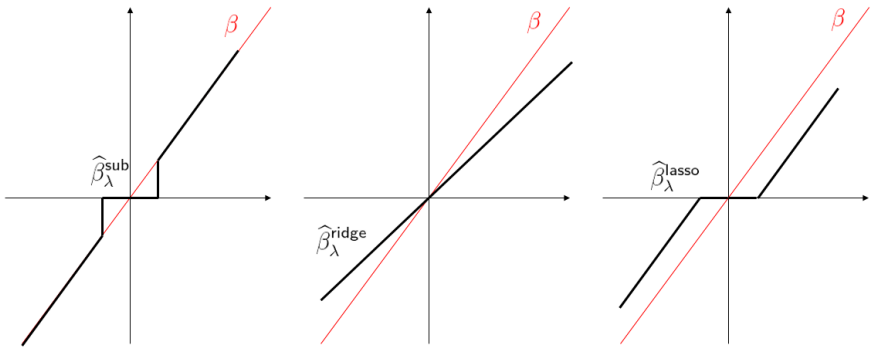


Figure 50.: Pénalisation basée sur la norme ℓ_0, ℓ_1 et ℓ_2 de $\boldsymbol{\beta}$, respectivement (inspiré de Hastie *et al.* (2016)).

3.3.6 *Optimisation et aspects algorithmiques*

En économétrie, l'optimisation (numérique) est devenue omniprésente dès que l'on a quitté le modèle Gaussien. Nous l'avions rapidement évoqué dans la section sur la famille exponentielle, et l'utilisation du score de Fisher (descente de gradient) pour résoudre la condition du premier ordre $\boldsymbol{X}^\top \boldsymbol{W}(\boldsymbol{\beta})^{-1}[\boldsymbol{y} - \widehat{\boldsymbol{y}}] = \boldsymbol{0}$. En apprentissage, l'optimisation est l'outil central. Et il est nécessaire d'avoir des algorithmes d'optimisation efficaces, pour résoudre des problèmes de la forme :

$$\widehat{\boldsymbol{\beta}} \in \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell(y_i, \beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\| \right\}.$$

Dans certains cas, au lieu de faire de l'optimisation globale, il suffit de considérer de l'optimisation par coordonnées (largement étudiée dans Daubechies and De Mol (2004)). Si $f : \mathbb{R}^d \to \mathbb{R}$ est convexe et différentiable, alors

$$\text{si } \boldsymbol{x} \text{ vérifie } f(\boldsymbol{x} + h\boldsymbol{e}_i) \geq f(\boldsymbol{x}) \text{ pour tout } h > 0 \text{ et } i \in \{1, \cdots, d\}, \text{ alors } f(\boldsymbol{x}) = \min\{f\},$$

où $\boldsymbol{e} = (\boldsymbol{e}_i)$ est la base canonique de $\mathbb{R}^d$. Cette propriété n'est toutefois pas vraie dans le cas non-différentiable. Mais si on suppose que la partie non-différentiable est séparable (additivement), elle redevient vraie. Plus précisément, si

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + \sum_{i=1}^{d} h_i(x_i) \text{ avec } \begin{cases} g : \mathbb{R}^d \to \mathbb{R} \text{ convexe-différentiable} \\ h_i : \mathbb{R} \to \mathbb{R} \text{ convexe.} \end{cases}$$

C'est le cas pour la régression LASSO, $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_{\ell_2} + \lambda\|\boldsymbol{\beta}\|_{\ell_1}$, comme le montre Tsen (2001). On peut alors utiliser un algorithme de descente par coordonnées: à partir d'une valeur initiale $\boldsymbol{x}^{(0)}$, on considère (en itérant)

$$x_j^{(k)} \in \text{argmin}\Big\{f(x_1^{(k)}, \cdots, x_{k-1}^{(k)}, x_k, x_{k+1}^{(k-1)}, \cdots, x_n^{(k-1)})\Big\} \text{ pour } j = 1, 2, \cdots, n.$$

Ces problèmes algorithmiques peuvent paraître secondaires à des économètres. Ils sont pourtant essentiels en apprentissage machine: une technique est intéressante s'il existe un algorithme stable et rapide, qui permet d'obtenir une solution. Ces techniques d'optimisation sont d'ailleurs transposables : par exemple, on pourra utiliser cette technique de descente par coordonnées dans le cas des méthodes SVM (dit « à support vecteur») lorsque l'espace n'est pas linéairement séparable, et qu'il convient de pénaliser l'erreur de classification (nous reviendrons sur cette technique dans la prochaine section).

3.3.7 *In-sample, out-of-sample et validation croisée*

Ces techniques semblent intellectuellement intéressantes, mais nous n'avons pas encore abordé le choix du paramètre de pénalisation λ. Mais ce problème est en fait plus général, car comparer deux paramètres $\widehat{\boldsymbol{\beta}}_{\lambda_1}$ et $\widehat{\boldsymbol{\beta}}_{\lambda_2}$ revient en fait à comparer deux modèles. En particulier, si on utilise une méthode de type LASSO, avec des seuils λ différents, on compare des modèles qui n'ont pas la même dimension. Dans la section 3.2.9, nous avions abordé le problème de la comparaison de modèles sous l'angle économétrique (en pénalisant les modèles trop complexes). Dans la littérature en apprentissage, juger de la qualité d'un modèle sur les données qui ont servi à le construire ne permet en rien de savoir comment le modèle se comportera sur des nouvelles données. Il s'agit du problème dit de « généralisation ». L'approche classique consiste alors à séparer l'échantillon (de taille n) en deux : une partie qui servira à entraîner le modèle (la base d'apprentissage, *in-sample*, de taille m) et une partie qui servira à tester le modèle (la base de test, *out-of-sample*, de taille $n-m$). Cette dernière permet alors de mesure un vrai risque prédictif. Supposons que les données soient générées par un modèle linéaire $y_i = \boldsymbol{x}_i^\intercal \boldsymbol{\beta}_0 + \varepsilon_i$ où les ε_i sont des réalisations de lois indépendantes et centrées. Le risque quadratique empirique *in-sample* est ici

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\Big([\boldsymbol{x}_i^\intercal \widehat{\boldsymbol{\beta}} - \boldsymbol{x}_i^\intercal \boldsymbol{\beta}_0]^2\Big) = \mathbb{E}\Big([\boldsymbol{x}_i^\intercal \widehat{\boldsymbol{\beta}} - \boldsymbol{x}_i^\intercal \boldsymbol{\beta}_0]^2\Big),$$

pour n'importe quelle observation i. En supposant les résidus ε Gaussiens, alors on peut montrer que ce risque vaut $\sigma^2 \text{trace}(\boldsymbol{\Pi}_{\mathcal{X}})/m$ soit $\sigma^2 p/m$. En revanche le risque quadratique empirique *out-of-sample* est ici

$$\mathbb{E}\Big([\boldsymbol{x}^\intercal \widehat{\boldsymbol{\beta}} - \boldsymbol{x}^\intercal \boldsymbol{\beta}_0]^2\Big)$$

où $\boldsymbol{x}$ est une nouvelle observation, indépendante des autres. On peut noter que

$$\mathbb{E}\Big([\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big|\boldsymbol{x}\Big)=\sigma^2\boldsymbol{x}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{x},$$

et en intégrant par rapport à $\boldsymbol{x}$,

$$\mathbb{E}\Big([\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big)=\mathbb{E}\Big(\mathbb{E}\Big([\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big|\boldsymbol{x}\Big)\Big)=\sigma^2\text{trace}\Big(\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\mathsf{T}]\mathbb{E}\Big[(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\Big]\Big).$$

L'expression est alors différente de celle obtenue *in-sample*, et en utilisation la majoration de Groves and Rothenberg (1969), on peut montrer que

$$\mathbb{E}\Big([\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big)\geq\sigma^2\frac{p}{m},$$

ce qui est assez intuitif, finalement. Hormis certains cas simple, il n'y a pas de formule simple. Notons toutefois que si $\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{0},\sigma^2\mathbb{I})$, alors $\boldsymbol{x}^\mathsf{T}\boldsymbol{x}$ suit une loi de Wishart, et on peut montrer que

$$\mathbb{E}\Big([\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_0]^2\Big)=\sigma^2\frac{p}{m-p-1}.$$

Si on regarde maintenant la version empirique: si $\widehat{\boldsymbol{\beta}}$ est estimé sur les m premières observations,

$$\widehat{\mathcal{R}}^{\mathsf{IS}}=\sum_{i=1}^{m}[y_i-\boldsymbol{x}_i^\mathsf{T}\widehat{\boldsymbol{\beta}}]^2\quad\text{et}\quad\widehat{\mathcal{R}}^{\mathsf{OS}}=\sum_{i=m+1}^{n}[y_i-\boldsymbol{x}_i^\mathsf{T}\widehat{\boldsymbol{\beta}}]^2,$$

et comme l'a noté Leeb (2008), $\widehat{\mathcal{R}}^{\mathsf{IS}}-\widehat{\mathcal{R}}^{\mathsf{OS}}\approx 2\cdot\nu$ où ν représente le nombre de degrés de libertés, qui n'est pas sans rappeler la pénalisation utilisée dans le critère d'Akaike.

La Figure 51 montre l'évolution respective de $\widehat{\mathcal{R}}^{\mathsf{IS}}$ et $\widehat{\mathcal{R}}^{\mathsf{OS}}$ en fonction de la complexité du modèle (nombre de degrés dans une régression polynomiale, nombre de noeuds dans des splines, etc). Plus le modèle est complexe, plus $\widehat{\mathcal{R}}^{\mathsf{IS}}$ va diminuer (c'est la courbe rouge). Mais ce n'est pas ce qui nous intéresse ici : on veut un modèle qui prédise bien sur de nouvelles données (autrement dit *out-of-sample*). Comme le montre la Figure 51, si le modèle est trop simple, il prédit mal (tout comme sur les données *in-sample*). Mais ce que l'on peut voir, c'est que si le modèle est trop complexe, on est dans une situation de « sur-apprentissage » : le modèle va commencer à modéliser le bruit.

Au lieu de séparer la base en deux, avec une partie des données qui vont servir à calibrer le modèle et une autre à étudier sa performance, il est aussi possible d'utiliser la validation croisée. Pour présenter l'idée générale, on peut revenir au « jackknife », introduit par Quenouille (1949) (et formalisé par Quenouille (1956) et Tukey (1958)) et utilisé en statistique pour réduire le biais. En effet, si on suppose que $\{y_1,\cdots,y_n\}$ est un échantillon tiré suivant une loi F_θ, et que l'on dispose d'un estimateur $T_n(\boldsymbol{y})=$

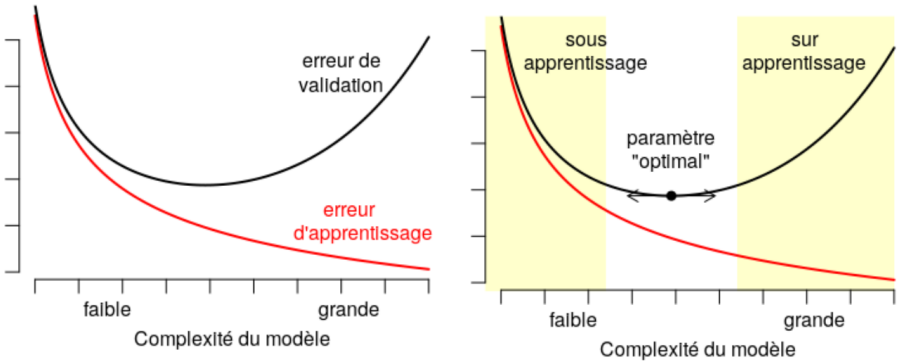


Figure 51.: Généralisation, et sur-apprentissage.

$T_n(y_1, \cdots, y_n)$, mais que cet estimateur est biaisé, avec $\mathbb{E}[T_n(\boldsymbol{Y})] = \theta + O\left(n^{-1}\right)$, il est possible de réduire le biais en considérant :

$$\widetilde{T}_n(\boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n} T_{n-1}(\boldsymbol{y}_{(i)}) \quad \text{avec} \quad \boldsymbol{y}_{(i)} = (y_1, \cdots, y_{i-1}, y_{i+1}, \cdots, y_n).$$

On peut alors montrer que $\mathbb{E}[\widetilde{T}_n(\boldsymbol{Y})] = \theta + O\left(n^{-2}\right)$.

L'idée de la validation croisée repose sur l'idée de construire un estimateur en enlevant une observation. Comme on souhaite construire un modèle prédictif, on va comparer la prévision obtenue avec le modèle estimé, et l'observation manquante :

$$\widehat{\mathcal{R}}^{\mathsf{CV}} = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \widehat{m}_{(i)}(\boldsymbol{x}_i))$$

On parlera ici de méthode « *leave-one-out* » (LOOCV).

On utilise classiquement cette technique pour trouver le paramètre optimal dans les méthodes de lissage exponentiel, pour des séries chronologiques. Dans le lissage simple, on va construire une prédiction de la forme ${}_t\widehat{y}_{t+1} = \alpha \cdot {}_{t-1}\widehat{y}_t + (1-\alpha)\cdot y_t$, avec $\alpha \in [0,1]$, et on va considérer :

$$\alpha^\star = \underset{\alpha\in[0,1]}{\operatorname{argmin}} \left\{ \sum_{t=2}^{T} \ell({}_{t-1}\widehat{y}_t, y_t) \right\},$$

comme le décrit Hyndman and Snyder (2009).

Le principal problème de la méthode « *leave-one-out* » est qu'elle nécessite de calibrer n modèles, ce qui peut être problématique en grande dimension. Une méthode alternative est la validation croisée par k-blocs (dit « *k-fold cross validation* ») qui consiste à utiliser

une partition de $\{1, \cdots, n\}$ en k groupes (ou blocs) de même taille, $\mathcal{I}_1, \cdots, \mathcal{I}_k$, et notons $\mathcal{I}_{\bar{j}} = \{1, \cdots, n\} \backslash \mathcal{I}_j$. En notant $\widehat{m}_{(j)}$ construit sur l'échantillon $\mathcal{I}_{\bar{j}}$, on pose alors :

$$\widehat{\mathcal{R}}^{k-\mathsf{CV}} = \frac{1}{k} \sum_{j=1}^{k} \mathcal{R}_j \quad \text{où} \quad \mathcal{R}_j = \frac{k}{n} \sum_{i \in \mathcal{I}_j} \ell(y_i, \widehat{m}_{(j)}(\boldsymbol{x}_i)).$$

La validation croisée standard, où une seule observation est enlevée à chaque fois (LOOCV), est un cas particulier, avec $k = n$. Utiliser $k = 5, 10$ a un double avantage par rapport à $k = n$: (1) le nombre d'estimations à effectuer est beaucoup plus faible, 5 ou 10 plutôt que n ; (2) les échantillons utilisés pour l'estimation sont moins similaires et donc, moins corrélés les uns aux autres, ce qui tend à éviter les excès de variance, comme le rappelle James and Tibshirani (2013).

Une autre alternative consiste à utiliser des échantillons boostrappés. Soit $\mathcal{I}_b$ un échantillon de taille n obtenu en tirant avec remise dans $\{1, \cdots, n\}$ pour savoir quelles observations $(y_i, \boldsymbol{x}_i)$ seront gardées dans la population d'apprentissage (à chaque tirage). Notons $\mathcal{I}_{\bar{b}} = \{1, \cdots, n\} \backslash \mathcal{I}_b$. En notant $\widehat{m}_{(b)}$ construit sur l'échantillon $\mathcal{I}_b$, on pose alors :

$$\widehat{\mathcal{R}}^{\mathsf{B}} = \frac{1}{B} \sum_{b=1}^{B} \mathcal{R}_b \quad \text{où} \quad \mathcal{R}_b = \frac{n_{\bar{b}}}{n} \sum_{i \in \mathcal{I}_{\bar{b}}} \ell(y_i, \widehat{m}_{(b)}(\boldsymbol{x}_i)),$$

où $n_{\bar{b}}$ est le nombre d'observations qui n'ont pas été conservées dans $\mathcal{I}_b$. On notera qu'avec cette technique, en moyenne $e^{-1} \sim 36.7\%$ des observations ne figurent pas dans l'échantillon boostrappé, et on retrouve un ordre de grandeur des proportions utilisées en créant un échantillon de calibration, et un échantillon de test. En fait, comme l'avait montré Stone (1977), la minimisation du AIC est à rapprocher du critère de validation croisée, et Shao (1997) a montré que la minimisation du BIC correspond à de la validation croisée de type k-fold, avec $k = n / \log n$.

3.4 QUELQUES OUTILS D'APPRENTISSAGE AUTOMATIQUE

3.4.1 *Réseaux de Neurones*

Les réseaux de neurones sont des modèles semi-paramétriques. Néanmoins, cette famille de modèles peut être appréhendée de la même manière que les modèles non-paramétriques: la structure des réseaux de neurones (présentée par la suite) peut être modifiée afin d'étendre la classe des fonctions utilisées pour approcher une variable d'intérêt. Plus précisément, Cybenko (1989) a démontré que l'ensemble des fonctions neuronales est dense dans l'espace des fonctions continues sur un compact. En d'autres termes, on a un cadre théorique permettant de garantir une forme d'approximation universelle. Il impose en outre une définition d'un neurone et met en avant l'existence d'un nombre de neurones suffisant pour approcher toute fonction continue sur un compact. Ainsi, un

phénomène continu peut être approché par une suite de neurones: on appellera cette suite « réseau de neurones à une couche ». Si ce théorème d'approximation universelle est démontré en 1989, le premier neurone artificiel fonctionnel fut introduit par Franck Rosenblatt au milieu du XXième siècle, dans Rosenblatt (1960). Ce neurone, qualifié de nos jours de « neurone élémentaire », porte le nom de « Perceptron ». Il a permis dans ses premières utilisations de déterminer le sexe d'un individu présenté au travers d'une photo. Si ce premier neurone est important, c'est qu'il introduit le premier formalisme mathématique d'un neurone biologique. On peut décrire un neurone artificiel par analogie avec une cellule nerveuse :

- les synapses apportant l'information à la cellule sont formalisées par un vecteur réel. La dimension du vecteur d'entrée du neurone (qui n'est d'autre qu'une fonction) correspond biologiquement au nombre de connections synaptiques;
- chaque signal apporté par une synapse est ensuite analysé par la cellule. Mathématiquement, ce schéma est transcrit par la pondération des différents éléments constitutifs du vecteur d'entrée;
- en fonction de l'information acquise, le neurone décide de retransmettre ou non un signal. Ce phénomène est répliqué par l'introduction d'une fonction d'activation. Le signal de sortie est modélisé par un nombre réel calculé comme image par la fonction d'activation du vecteur d'entrée pondéré.

Ainsi, un neurone artificiel est un modèle semi-paramétrique. Le choix de la fonction d'activation est en effet laissé à l'utilisateur. Nous introduisons dans le paragraphe qui suit une formalisation rigoureuse qui nous permettra de poser le modèle, et de faire le lien avec les notations économétriques usuelles. On peut alors définir un neurone élémentaire formellement par :

- un espace d'entrée $\mathcal{X}$, généralement $\mathbb{R}^k$ avec $k \in \mathbb{N}^*$;
- un espace de sortie $\mathcal{Y}$, généralement $\mathbb{R}$ ou un ensemble fini (classiquement $\{0,1\}$, mais on préférera ici $\{-1,+1\}$);
- un vecteur de paramètres $\boldsymbol{w} \in \mathbb{R}^p$
- une fonction d'activation $\phi : \mathbb{R} \to \mathbb{R}$. Cette fonction doit être dans l'idéal monotone, dérivable et bornée (on dira ici « saturante ») afin de s'assurer de certaines propriétés de convergence.

Cette dernière fonction ϕ fait penser aux transformations logistique ou probit, populaires en économétrie (qui sont des fonctions de répartition, à valeur dans $[0,1]$, idéal quand $\mathcal{Y}$ est l'ensemble $\{0,1\}$). Pour les réseaux de neurones, on utilisera plutôt la tangente

hyperbolique, la fonction arc-tangente ou les fonctions sigmoïdes pour des problèmes de classification sur $\mathcal{Y} = \{-1, +1\}$ (ces dernières évoqueront la transformation logistique de économètres). On appellera neurone toute application f_w de $\mathcal{X}$ dans $\mathcal{Y}$ définie par :

$$y = f_{\boldsymbol{w}}(\boldsymbol{x}) = \phi(\boldsymbol{w}^\top \boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

Pour le Perceptron introduit par Rosenblatt (1960), on assimile un neurone élémentaire à la fonction :

$$y = f_{\boldsymbol{w}}(\boldsymbol{x}) = \text{signe}(\boldsymbol{w}^\top \boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathcal{X}$$

On remarque que selon cette formalisation, beaucoup de modèles statistiques comme par exemple les régressions logistiques pourraient être vus comme des neurones. En effet si l'on regarde d'un peu plus près, tout modèle GLM (« *Generalized Linear Model* ») pourrait s'interpréter comme un neurone formel où la fonction d'activation ϕ n'est d'autre que l'inverse de la fonction de lien canonique (par exemple). Si g désigne la fonction de lien du GLM, $\boldsymbol{w}$ le vecteur de paramètres, y la variable à expliquer et $\boldsymbol{x}$ le vecteur des variables explicatives de même dimension que $\boldsymbol{w}$:

$$g(\mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]) = \boldsymbol{w}^\top \boldsymbol{x}$$

On retrouve la modélisation neuronale en prenant $\phi = g^{-1}$. Cependant, là où réside la différence majeure entre les GLM et le modèle neuronale est que ce dernier n'introduit aucune hypothèse de distribution sur $Y|\boldsymbol{X}$ (on n'a d'ailleurs pas besoin d'introduire ici de modèle probabiliste). D'autre part, lorsque le nombre de neurones par couche augmente, la convergence n'est pas nécessairement garantie si la fonction d'activation ne vérifie pas certaines propriétés (qu'on ne retrouve pas dans la majorité des fonctions de liens canoniques des GLM). Cependant, comme énoncé précédemment, la théorie des réseaux de neurones introduit des contraintes mathématiques supplémentaires sur la fonction g (détaillé dans Cybenko (1989)). Ainsi par exemple, une régression logistique peut être perçue comme un neurone alors que les régressions linéraires généralisées ne vérifient pas toutes les hypothèses nécessaires.

Toujours par analogie avec le fonctionnement du système nerveux, il est alors possible de connecter différents neurones entre eux. On parlera de structure de réseaux de neurones par couche. Chaque couche de neurones recevant à chaque fois le même vecteur d'observation.

Pour revenir à une analogie plus économétrique, on peut imaginer passer par une étape intermédiaire (on reviendra sur cette construction dans la Figure 52), par exemple en ne faisant pas une régression sur les variables brutes $\boldsymbol{x}$ mais un ensemble plus faible de variables orthogonales, obtenues par exemple suite à une analyse en composantes principales. Soit $\boldsymbol{A}$ la matrice associée à cette transformation linéaire, avec $\boldsymbol{A}$ de taille $k \times p$ si on souhaite utiliser les p premières composantes. Notons $\boldsymbol{z}$ la transformation de

$\boldsymbol{x}$, au sens où $\boldsymbol{z} = \boldsymbol{A}^\top \boldsymbol{x}$, ou encore $z_j = \boldsymbol{A}_j^\top \boldsymbol{x}$. Un généralisation du modèle précédent peut être de poser

$$y = f(x) = \phi(\boldsymbol{w}^\top \boldsymbol{z}) = \phi(\boldsymbol{w}^\top \boldsymbol{A}^\top \boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{X},$$

où cette fois $\boldsymbol{w} \in \mathbb{R}^p$. On a ici une transformation linéaire (en considérant une analyse en composante principale) mais on peut imaginer une généralisation avec des transformée non-linéaire, avec une fonction de la forme

$$y = f(x) = \phi(\boldsymbol{w}^\top F_{\boldsymbol{A}}(\boldsymbol{x})), \quad \forall \boldsymbol{x} \in \mathcal{X},$$

où F est ici une fonction $\mathbb{R}^k \to \mathbb{R}^p$. C'est le réseau de neurone à deux couches. Plus généralement, pour formaliser la construction, on introduit les notations suivantes :

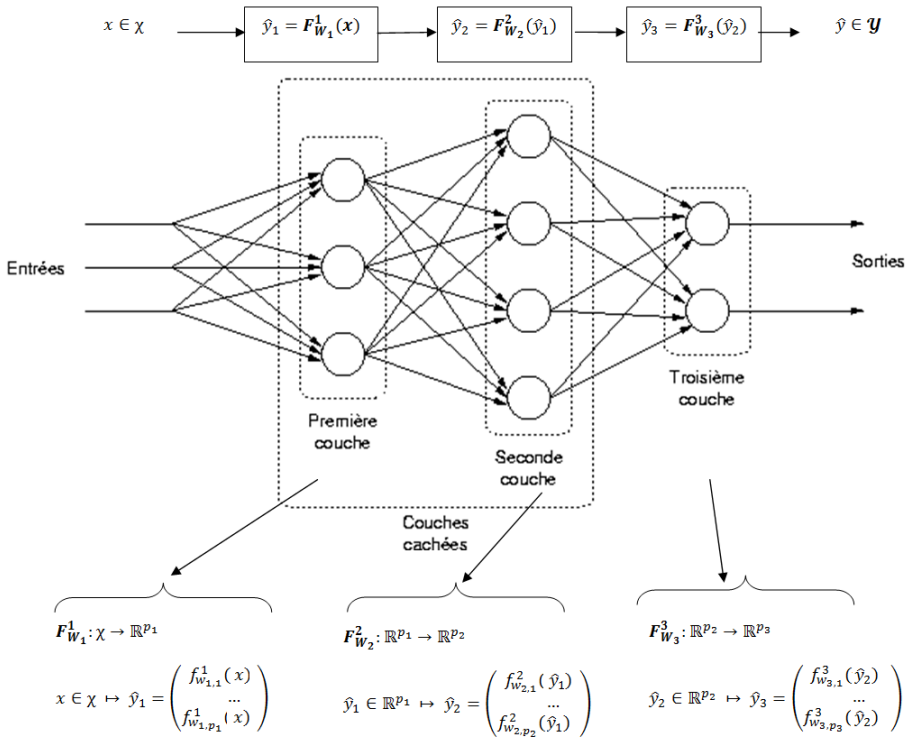
- $K \in \mathbb{N}^*$: nombre de couches;
- $\forall k \in \{1, \cdots K\}$, p_k représente le nombre de neurones dans la couche k;
- $\forall k \in \{1, \cdots K\}$, W_k désigne la matrice des paramètres associés à la couche k. Plus précisément, W_k est une matrice $p_k \times p_{k-1}$ et pour tout $l \in \{1, \cdots p_k\}$, $w_{k,l} \in \mathbb{R}^{p_{k-1}}$ désigne le vecteur de poids associé au neurone élémentaire l de la couche k;
- on appellera $W = \{W_1, .., W_K\}$, l'ensemble des paramètres associés au réseau de neurones.
- $F_{W_k}^k : \mathbb{R}^{p_{k-1}} \to \mathbb{R}^{p_k}$ désigne la fonction de transfert associé à la couche k. Pour des raisons de simplification, on pourra également écrire F^k;
- $\hat{y}_k \in \mathbb{R}^{p_k}$ représentera le vecteur image de la couche $k \in \{1, \cdots, K\}$;
- on appellera $F = F_W = F^1 \circ \cdots \circ F^K$ la fonction de transfert associée au réseau global. A ce titre, si $\boldsymbol{x} \in \mathcal{X}$, on pourra noter $\widehat{\boldsymbol{y}} = F_W(\boldsymbol{x})$.

La Figure 52 permet d'illustrer les notations présentées ici[8]. Chaque cercle représente un neurone élémentaire. Chaque rectangle englobant plusieurs cercles représente une couche. On parle de couche d'entrée pour la première couche prenant en « input » les observation $\boldsymbol{x} \in \mathcal{X}$, de couche de sortie pour la couche fournissant en « output » la prédiction $\hat{\boldsymbol{y}} \in \mathcal{Y}$. Les autres couches sont couramment appelées couches cachées.

Un réseau de neurones multicouches est donc également un modèle semi-paramétrique dont les paramètres sont l'ensemble des composantes des matrices W_k pour tout entier k de $\{1, \cdots, K\}$. Chaque fonction d'activation associée à chaque neurone (chaque cercle de la Figure 52) est à déterminer par l'utilisateur.

8 Source: http://intelligenceartificielle.org.

Figure 52.: Exemple de notations associées aux réseaux de neurones multicouche.



Une fois que les paramètres à calibrer du modèle sont identifiés (ici les réels constituant les matrices W_k pour chaque couche $k \in \{1, \cdots, K\}$) , il est nécessaire de fixer une fonction de perte ℓ. En effet, on rappelle que l'objectif de l'apprentissage supervisé sur une base d'apprentissage de $n \in \mathbb{N}^*$ couples $(y_i, \boldsymbol{x}_i) \in \mathcal{Y} \times \mathcal{X}$ est de minimiser le risque empirique :

$$\widehat{\mathcal{R}}_n(F_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, F_W(\boldsymbol{x}_i))$$

Afin d'illustrer les propos, intéressons nous à l'exemple suivant qui illustrera également la démarche opérée. Supposons que nous observons un phénomène y aux travers de n observations $y_i \in [-1, 1]$. On souhaiterait expliquer ce phénomène à partir des variables explicatives $\boldsymbol{x}$ que l'on suppose à valeurs réelles. La « théorie de l'approximation universelle » nous indique qu'un réseau à une couche de neurones devrait permettre de modéliser le phénomène (sous hypothèse qu'il soit continue). On note toutefois que ce théorème ne donne pas de vitesse de convergence. Il est alors laissé à l'utilisateur le choix de la structure. Ainsi par exemple une première structure pourrait être un simple neurone dont la fonction d'activation serait la fonction tangente hyperbolique.

On aurait ainsi comme premier modèle:

$$y_1 = \tanh(w_0 + w_1 x)$$

où les paramètres $w_0,\ w_1$ sont les paramètres à optimiser de sorte que sur les données d'apprentissage, le risque empirique soit minimal.

Si l'on suit toujours la philosophie du théorème d'approximation universelle, en ajoutant plusieurs neurones, l'erreur est censée diminuer. Cependant, ne connaissant pas la fonction à estimer, on ne peut l'observer qu'aux travers de l'échantillon. Ainsi, mécaniquement, on s'attend à ce que plus on ajoute de paramètres, plus l'erreur sur la base d'apprentissage diminue. L'analyse de l'erreur sur la base de test permet alors d'évaluer notre capacité à généraliser (cf partie précédente).

On peut ainsi s'intéresser à un second modèle qui cette fois utilise plusieurs neurones. Par exemple, considérons le modèle

$$y_2 = w_a \tanh(w_0 + w_1 x) + w_b \tanh(w_2 + w_3 x) + w_c \tanh(w_4 + w_5 x)$$

où les paramètres $w_0, .., w_5$ ainsi que w_a, w_b, w_c sont les paramètres à optimiser. Calibrer un réseaux de neurones revient alors à réitérer ces étapes de modification de la structure jusqu'à minimisation du risque sur la base de test.

Pour une structure de réseau de neurones fixée (c'est-à-dire nombre de couches, nombre de neurones par couches et fonctions d'activation fixés), le programme revient donc à déterminer l'ensemble de paramètres $W^* = (W_1, ..., W_K)$ de sorte que :

$$W^* \in \underset{W=(W_1,...,W_K)}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, F_W(\boldsymbol{x}_i)) \right\}.$$

De cette formule apparaît l'importance du choix de la fonction ℓ. Cette fonction de perte quantifie l'erreur moyenne commise par notre modèle F_W sur la base d'apprentissage. *A priori* ℓ peut être choisie arbitrairement. Cependant, dans l'optique de résoudre un programme d'optimisation, on préféra des fonctions de coût sous-différentiables et convexes afin de garantir la convergence des algorithmes d'optimisation. Parmi les fonctions de perte classiques, en plus de la fonction de perte quadratique ℓ_2 on retiendra la fonction dite « *Hinge* » - $\ell(y, \hat{y}) = \max(0, 1 - y\hat{y})$ - ou la fonction dite logistique - $\ell(y, \hat{y}) = \log(1 - e^{-y\hat{y}})$.

En définitive les réseaux de neurones sont des modèles semi-paramétriques dont le nombre de paramètres est croissant avec le nombre de couches et de neurones par couche. Il est laissé à l'utilisateur de choisir les fonctions d'activation et la structure du réseau. Ceci explique l'analogie avec la philosophie des modèles non-paramétriques faite auparavant.

Les réseaux de neurones ont été utilisés très tôt en économie et en finance, en particulier sur les défauts d'entreprises - Tam and Kiang (1992) ou Altman and Varetto (1994) - ou plus récemment la notation de crédit - Blanco and Rayo (2013) ou Khashman

(2011). Cependant les structures telles que présentées précédemment sont généralement limitées. L'apprentissage profond (ou « *deep learning* ») caractérise plus particulièrement des réseaux de neurones plus complexes (parfois plus d'une dizaine de couches avec parfois des centaines de neurones par couche). Si aujourd'hui ces structures sont très populaires en analyse du signal (image, texte, son) c'est qu'elles sont capables à partir d'une quantité d'observations très importante d'extraire des informations que l'humain ne peut percevoir et de faire face à des problèmes non linéaires, comme le rappelle Bengio and Hinton (2015).

L'extraction d'informations peut, par exemple, se faire grâce à la convolution. Procédé non supervisé, il a permis notamment d'obtenir d'excellentes performances dans l'analyse d'image. Techniquement, cela peut s'apparenter à une transformation à noyaux (comme utilisé dans les techniques SVM). Si une image peut être perçue comme une matrice dont chaque coordonnée représente un pixel, une convolution reviendrait à appliquer une transformation sur un point (ou une zone) de cette matrice générant ainsi une nouvelle donnée. Le procédé peut ainsi être répété en appliquant des transformations différentes (d'où la notion de couches convolutives). Le vecteur final obtenu peut alors enfin alimenter un modèle neuronal comme introduit dans le paragraphe précédent. En fait, plus généralement, une couche de convolution peut être perçue comme un filtre qui permet de transformer la donnée initiale.

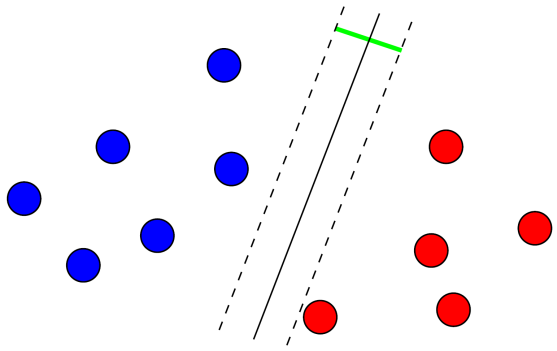
Une explication intuitive pour laquelle l'apprentissage approfondi, en particulier les réseaux profonds, est si puissant pour décrire des relations complexes dans les données, c'est leur construction autour de l'approximation fonctionnelle simple et l'exploitation d'une forme de hiérarchie, comme le note Lin and Rolnick (2016). Néanmoins les modèles de type « *deep learning* » sont plus difficiles à appréhender car ils nécessitent beaucoup de jugement empirique. En effet, si aujourd'hui les bibliothèques open sources (keras, torch, etc.) permettent de paralléliser plus facilement les calculs en utilisant par exemple les GPU (« *Graphical Processor Units* »), il reste néanmoins à l'utilisateur de déterminer la structure du réseau de neurones le plus approprié. Cette extension des réseaux de neurones est étudiée dans le Chapitre 4.

3.4.2 *Support Vecteurs Machine*

Comme nous l'avions noté auparavant, dans les problèmes de classification en apprentissage machine (comme en traitement du signal) on préférera avoir des observations dans l'ensemble $\{-1, +1\}$ (plutôt que $\{0, 1\}$, comme en économétrie). Avec cette notation, Cortes and Vapnik (1995) ont posé les bases théoriques des modèles dits SVM, proposant une alternative aux réseaux de neurones alors très populaires comme algorithme de classification dans la communauté de l'apprentissage machine. L'idée initiale des méthodes de « *Support Vector Machine* » (SVM) consiste à trouver un hyperplan séparateur divisant l'espace en deux ensembles de points le plus homogène possible (i.e. contenant des labels identiques). En dimension deux, l'algorithme consiste à déterminer une droite séparant

l'espace en deux zones les plus homogènes possibles. La résolution de ce problème possédant parfois une infinité de solution (il peut en effet exister une infinité de droites qui séparent l'espace en deux zones distinctes et homogènes), on rajoute généralement une contrainte supplémentaire. L'hyperplan séparateur doit se trouver le plus éloigné possible des deux sous-ensembles homogènes qu'il engendre. On parlera ainsi de marge. L'algorithme ainsi décrit est alors un SVM linéaire à marge.

Figure 53.: Schéma d'illustration d'un SVM à marge, Vert (2017).



Si un plan peut être caractérisé entièrement par un vecteur directeur $\boldsymbol{w}$ orthogonal à ce dernier et une constante b, appliquer un algorithme SVM à un ensemble de $n \in \mathbb{N}^*$ points $\boldsymbol{x}_i$ de $\mathbb{R}^p$ labellisés par $y_i \in \{-1, 1\}$ revient alors à résoudre un programme d'optimisation sous contrainte similaire à celui d'un LASSO (distance quadratique sous contrainte linéaire). Plus particulièrement, on sera amené à résoudre :

$$(\boldsymbol{w}^\star, b^\star) = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \left\{ \|\boldsymbol{w}\|^2 \right\} = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \left\{ \boldsymbol{w}^\mathsf{T}\boldsymbol{w} \right\},$$

$$\text{sous contrainte } \ \forall i \in \{1, \cdots, n\}, \ \begin{cases} \boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b \geq +1 & \text{lorsque} \quad y_i = +1 \\ \boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b \leq -1 & \text{lorsque} \quad y_i = -1 \end{cases}$$

La contrainte peut être relâchée en autorisant que dans un sous-ensemble, un point puisse ne pas être du même label que la majeure partie des points de ce sous-ensemble à condition de ne pas être trop loin de la frontière. C'est ce qu'on appelle les SVM linéaire à marge légère (« *soft margin* »). De manière heuristique, comme en pratique, bien souvent, on ne peut pas avoir $y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) - 1 \geq 0$ pour tout $i \in \{1, \cdots, n\}$, on relâche en introduisant des variables positives $\boldsymbol{\xi}$ telle que

$$\begin{cases} \boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b \geq +1 - \xi_i & \text{lorsque} \quad y_i = +1 \\ \boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b \leq -1 + \xi_i & \text{lorsque} \quad y_i = -1 \end{cases} \tag{17}$$

avec $\xi_i \geq 0$. On a une erreur de classification si $\xi_i > 1$, et on va alors introduire une pénalité, un coût à payer pour chaque erreur commise. On cherche alors à résoudre un problème quadratique

$$\min\left\{\frac{1}{2}\boldsymbol{\omega}^\top\boldsymbol{\omega} + C\mathbf{1}^\top\mathbf{1}_{\boldsymbol{\xi}>1}\right\}$$

sous la contrainte (17), qui pourra être résolu de manière numérique très efficacement par descente de coordonnées (décrit auparavant).

S'il n'est pas possible de séparer les points, une autre astuce possible consiste à les transformer dans une dimension supérieure, de sorte que les données deviennent alors linéairement séparables. Trouver la bonne transformation qui sépare les données est toutefois très difficile. Cependant, il existe une astuce mathématique pour résoudre ce problème avec élégance, en définissant les transformations $T(\cdot)$ et les produits scalaires via un noyau $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle T(\boldsymbol{x}_1), T(\boldsymbol{x}_2)\rangle$. L'un des choix les plus courants pour une fonction de noyau est la fonction de base radiale (noyau gaussien) $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\Big(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2)\|^2\Big)$. Il n'existe néanmoins pas de règles à ce jour permettant de choisir le « meilleur » noyau. Comme mentionné au début de la section précédente, cette technique est basé sur de la minimisation de distance, et il n'a aucune prévision de la probabilité d'être positif ou négatif (mais une interprétation probabiliste est néanmoins possible, comme le montre Grandvalet and Bengio (2005), par exemple).

3.4.3 *Arbres, Bagging et Forêts Aléatoires*

Les arbres de classification ont été introduits dans Quinlan (1956) mais c'est surtout Breiman (2001a) qui a assuré la popularité de l'algorithme. On parle de modèle CART pour « *Classification And Regression Tree* ». L'idée est de diviser consécutivement (par une notion de branchement) les données d'entrée jusqu'à ce qu'un critère d'affectation (par rapport à la variable cible) soit atteint, selon une règle prédéfinie.

L'intuition de la construction des arbres de classification est la suivante. L'entropie $H(\boldsymbol{x})$ est associée à la quantité de désordre dans les données $\boldsymbol{x}$ par rapport aux modalités prises par la variable de classification y, et chaque partition vise à réduire ce désordre. L'interprétation probabiliste est de créer les groupes les plus homogènes possible, en réduisant la variance par groupe (variance intra), ou de manière équivalente en créant deux groupes aussi différents que possible, en augmentant la variance entre les groupe (variance inter). à chaque étape, nous choisissons la partition qui donne la plus forte réduction de désordre (ou de variance). L'arbre de décision complet se développe en répétant cette procédure sur tous les sous-groupes, où chaque étape k aboutit à une nouvelle partition en 2 branches, qui subdivise notre ensemble de données en 2. Enfin, on décide quand mettre fin à cette constitution de nouvelles branches, en procédant à des affectations finales (nœuds dits foliaires). Il existe plusieurs options pour mettre fin à cette croissance. L'une est de construire un arbre jusqu'à ce que toutes les feuilles

soient pures, c'est à dire composées d'une seule observation. Une autre option est de définir une règle d'arrêt liée à la taille, ou à la décomposition, des feuilles. Les exemples de règles d'arrêt peuvent être d'une taille minimale (au moins 5 éléments par feuille), ou une entropie minimale. On parlera alors d'élagage de l'arbre: on laisse l'arbre grossir, puis on coupe certaines branches *a posteriori* (ce qui est différent de l'introduction d'un critère d'arrêt *a priori* au processus de croissance de l'arbre - par exemple en imposant une taille minimale aux feuilles, ou d'autres critères discutés dans Breiman (2001a)).

à un nœud donné, constitué de n_0 observations $(\boldsymbol{x}_i, y_i)$ avec $i \in \mathcal{I}_0$, on va couper en deux branches (une à gauche et une à droite), partitionnant ainsi $\mathcal{I}_0$ en $\mathcal{I}_\text{g}$ et $\mathcal{I}_\text{d}$. Soit I le critère d'intérêt, comme l'entropie du nœud (ou plutôt du nœud vu en tant que feuille):

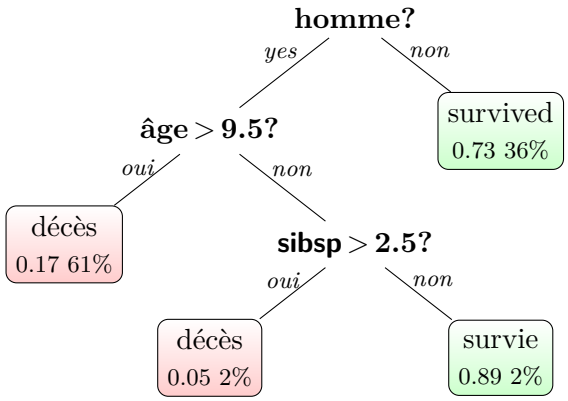
$$I(\boldsymbol{y}_0) = -n_0 p_0 \log p_0 \quad \text{où} \quad p_0 = \frac{1}{n_0} \sum_{i \in \mathcal{I}_0} y_i,$$

ou la variance du nœud:

$$I(\boldsymbol{y}_0) = n_0 p_0 (1 - p_0) \quad \text{où} \quad p_0 = \frac{1}{n_0} \sum_{i \in \mathcal{I}_0} y_i,$$

ce dernier étant également l'indice d'impureté de Gini.

Figure 54.: Schéma d'illustration d'un arbre de décision permettant de prédire le taux de survie d'un individu du Titanic.



On partitionnera entre la branche gauche et la branche droite si le gain $I(\boldsymbol{y}_0) - \Big[I(\boldsymbol{y}_\text{g}) + I(\boldsymbol{y}_\text{d})\Big]$ est suffisamment important. Lors de la construction des arbres, on va chercher la partition qui donne le gain le plus important possible. Ce problème combinatoire étant complexe, le critère suggéré par Breiman (2001a) est de considérer un découpage suivant une des variables, avec $\mathcal{I}_\text{g} = \{i \in \mathcal{I}_0 : x_{k,i} < s\}$ et $\mathcal{I}_\text{d} = \{i \in \mathcal{I}_0 : x_{k,i} > s\}$, pour une variable k et un seuil s (si la variable est continue, sinon on considère des regroupements de modalités pour des variables qualitatives).

Les arbres de décision ainsi décrits sont simples à obtenir et faciles à interpréter (comme le montre la Figure 54 sur les données du Titanic[9]), mais ils sont peu robustes, et leur pouvoir prédictif est souvent très faible, en particulier si l'arbre est très profond. Une idée naturelle est de développer un ensemble de modèles d'arbres à peu près indépendants, qui prédisent conjointement mieux qu'un modèle d'arbre unique. On va utiliser le bootstrap, en tirant (avec remise) n observations parmi $\{(\boldsymbol{x}_i, y_i)\}$. Ces ensembles d'arbres - naturellement appelés « forêts » - une fois agrégés donnent souvent de bien meilleurs résultats que les arbres isolés, mais ils sont difficiles à interpréter. Ces techniques ressemblent toutefois beaucoup à ce qui est fait lorsque l'on utilise les techniques de bootstrap en régression (par exemple pour construire des tubes de confiance dans une régression fonctionnelle).

Le principe du « *bagging* », pour « *bootstrap aggregating* », consiste à générer des échantillons aléatoires, en tirant avec remise dans l'échantillon d'origine, comme pour le bootstrap. Chaque échantillon ainsi généré permet d'estimer un nouvel arbre de classification, formant ainsi une forêt d'arbres. C'est l'agrégation de tous ces arbres qui conduit à la prévision. Le résultat global est moins sensible à l'échantillon initial et donne souvent de meilleurs résultats de prévision.

Les forêts aléatoires, ou « *random forests* » reposent sur le même principe que le « *bagging* », mais en plus, lors de la construction d'un arbre de classification, à chaque branche, un sous-ensemble de m covariables est tiré aléatoirement. Autrement dit, chaque branche d'un arbre ne s'appuie pas sur le même ensemble de covariables. Cela permet d'amplifier la variabilité entre les différents arbres et d'obtenir, au final, une forêt composée d'arbres moins corrélés les uns aux autres.

3.4.4 *Sélection de modèle de classification*

étant donné un modèle $m(\cdot)$ approchant $\mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$, et un seuil $s \in [0, 1]$, posons

$$\widehat{y}^{(s)} = \mathbf{1}[m(\boldsymbol{x}) > s] = \begin{cases} 1 \text{ si } m(\boldsymbol{x}) > s \\ 0 \text{ si } m(\boldsymbol{x}) \leq s \end{cases}$$

La matrice de confusion est alors le tableau de contingence associé aux comptages $\boldsymbol{N} = [N_{u,v}]$ avec

$$N_{u,v}^{(s)} = \sum_{i=1}^{n} \mathbf{1}(\widehat{y}_i^{(s)} = u, y_j = v)$$

pour $(u, v) \in \{0, 1\}$. La Table 13 présente un tel tableau, avec le nom de chacun des éléments : TP (*true positive*) sont les vrais positifs, correspondant aux 1 prédits en 1, TN (*true negative*) sont les vrais négatifs, correspondant aux 0 prédits en 0, FP (*false*

9 Ce jeu de données, contenant des informations sur tous les passagers (et membres d'équipage) du Titanic, dont la variable y indiquant si la personne a survécu a été abondamment utilisé pour illustrer les techniques de classification, voir https://www.kaggle.com/c/titanic/data.

	$y=0$	$y=1$	
$\widehat{y}_s=0$	TN_s	FN_s	TN_s+FN_s
$\widehat{y}_s=1$	FP_s	TP_s	FP_s+TP_s
	TN_s+FP_s	FN_s+TP_s	n

Table 13.: Matrice de confusion, ou tableau de contingence pour un seuil s donné.

positive) sont les faux positifs, correspondant aux 0 prédits en 1, et enfin FN (*false negative*) sont les faux négatifs, correspondant aux 1 prédits en 0).

Plusieurs quantités sont dérivées de ce tableau. La sensibilité correspond à la probabilité de prédire 1 dans la population des 1, ou taux de vrais positifs. La spécificité est la probabilité de prédire 0 dans la population des 0 ou taux de vrais négatifs. On s'intéressera toutefois davantage au taux de faux négatifs, c'est à dire la probabilité de prédire 1 dans la population des 0. La représentation de ces deux valeurs lorsque s varie donne la courbe ROC (« *receiver operating characteristic* ») :

$$\text{ROC}_s = \left(\frac{\text{FP}_s}{\text{FP}_s+\text{TN}_s}, \frac{\text{TP}_s}{\text{TP}_s+\text{FN}_s}\right) = (\mathsf{sensibility}_s, 1-\mathsf{specificity}_s) \text{ pour } s\in[0,1].$$

Une telle courbe est présentée dans la partie suivante, sur des données réelles.

Les deux grandeurs intensivement utilisées en apprentissage automatique sont l'indice κ, qui compare la précision observée avec celle espérée, avec un modèle aléatoire (tel que décrit dans Landis and Koch (1977)) et l'AUC correspondant à l'aire sous la courbe ROC. Pour le premier indice, une fois choisi s, notons $\boldsymbol{N}^{\perp}$ le tableau de contingence correspond aux cas indépendants (défini à partir de $\boldsymbol{N}$ dans le test d'indépendance du chi-deux). On pose alors

$$\text{précision totale} = \frac{\text{TP+TN}}{n}$$

alors que

$$\text{précision aléatoire} = \frac{[\text{TN+FP}]\cdot[\text{TP+FN}]+[\text{TP+FP}]\cdot[\text{TN+FN}]}{n^2}$$

On peut alors définir

$$\kappa = \frac{\text{précision totale} - \text{précision aléatoire}}{1-\text{précision aléatoire}}$$

Classiquement s sera fixé égal à 0.5, comme dans une classification Bayésienne naïve, mais d'autres valeurs peuvent être retenues, en particulier si les deux erreurs ne sont pas symétriques (nous reviendrons sur ce point dans un exemple par la suite).

Il existe des compromis entre des modèles simples et complexes mesurés par leur nombre de paramètres (ou plus généralement les degrés de liberté) en matière de performance

et de coût. Les modèles simples sont généralement plus faciles à calculer, mais peuvent conduire à des ajustements plus mauvais (avec un biais élevé par exemple). Au contraire, les modèles complexes peuvent fournir des ajustements plus précis, mais risquent d'être coûteux en termes de calcul. En outre, ils peuvent surpasser les données ou avoir une grande variance et, tout autant que des modèles trop simples, ont de grandes erreurs de test. Comme nous l'avons rappelé auparavant, dans l'apprentissage machine, la complexité optimale du modèle est déterminée en utilisant le compromis de biais-variance.

3.4.5 *De la classification à la régression*

Comme nous l'avons rappelé en introduction, historiquement, les méthodes d'apprentissage automatique se sont orientées autour des problèmes de classification (avec éventuellement plus de 2 modalités[10]), et assez peu dans le cas où la variable d'intérêt y est continue. Néanmoins, il est possible d'adapter quelques techniques, comme les arbres et les forêts aléatoires, le boosting, ou les réseaux de neurones.

Pour les arbres de régression, Morgan and Sonquist (1963) ont proposé la méthode AID, basée sur la formule de décomposition de la variance de l'équation (10), avec un algorithme proche de celui de la méthode CART décrite auparavant. Dans le contexte de la classification, on calculait, à chaque nœud (dans le cas de l'indice d'impureté de Gini) en sommant sur la feuille de gauche $\{x_{k,i} < s\}$ et celle de droite $\{x_{k,i} > s\}$

$$I = \sum_{i:x_{k,i}<s} \overline{y}_{\mathrm{g}}\Big(1-\overline{y}_{\mathrm{g}}\Big) + \sum_{i:x_{k,i}>s} \overline{y}_{\mathrm{d}}\Big(1-\overline{y}_{\mathrm{d}}\Big)$$

où $\overline{y}_{\mathrm{g}}$ et $\overline{y}_{\mathrm{d}}$ désignent les fréquences de 1 dans la feuille de gauche et de droite, respectivement. Dans le cas d'un arbre de régression, on utilisera

$$I = \sum_{i:x_{k,i}<s} \Big(y_i-\overline{y}_{\mathrm{g}}\Big)^2 + \sum_{i:x_{k,i}>s} \Big(y_i-\overline{y}_{\mathrm{d}}\Big)^2$$

qui va correspondre à la somme (pondérée) des variances intra. Le partage optimal sera celui qui aura le plus de variance intra (on veut les feuilles les plus homogènes possibles) ou de manière équivalente, on veut maximiser la variance intra.

Dans le contexte des forêts aléatoires, on utilise souvent un critère majoritaire en classification (la classe prédite sera la classe majoritaire dans une feuille), alors que pour la régression, on utilise la moyenne des prédictions, sur tous les arbres.

Dans la partie précédente, nous avons présenté la dimension « apprentissage » de l'apprentissage automatique en présentant le « *boosting* ». Dans un contexte de régression

10 Par exemple dans le cas de reconnaissance de lettres ou de chiffres.

(variable y continue), l'idée est de créer une succession de modèles en écrivant l'équation (13) sous la forme :

$$m^{(k)}(\boldsymbol{x}) = m^{(k-1)}(\boldsymbol{x}) + \alpha_k \underset{h\in\mathcal{H}}{\text{argmin}} \left\{ \sum_{i=1}^{n} (y_i, m^{(k-1)}(\boldsymbol{x}) + h(\boldsymbol{x}))^2 \right\}$$

où α_k est un paramètre de « *shrinkage* », où le second terme correspond à un arbre de régression, sur les résidus, $y_i - m^{(k-1)}(\boldsymbol{x}_i)$.

Mais il existe d'autres techniques permettant d'apprendre de manière séquentielle. Dans un modèle additif (GAM) on va chercher une écriture de la forme

$$m(\boldsymbol{x}) = \sum_{j=1}^{p} m_j(x_j) = m_1(x_1) + \cdots + m_p(x_p)$$

L'idée de la poursuite de projection repose sur une décomposition non pas sur les variables explicatives, mais sur des combinaisons linéaires. On va ainsi considérer un modèle

$$m(\boldsymbol{x}) = \sum_{j=1}^{k} g_j(\boldsymbol{\omega}_j^\intercal \boldsymbol{x}) = g_1(\boldsymbol{\omega}_1^\intercal \boldsymbol{x}) + \cdots + g_k(\boldsymbol{\omega}_k^\intercal \boldsymbol{x}).$$

Tout comme les modèles additifs, les fonctions $g_1, \cdots, g_k$ sont à estimer, tout comme les directions $\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_k$. Cette écriture est relativement générale, et permet de tenir compte d'interactions et d'effets croisés (ce que nous ne pouvions pas faire avec les modèles additifs qui ne tiennent compte que de non-linéarités). Par exemple en dimension 2, un effet multiplicatif $m(x_1, x_2) = x_1 \cdot x_2$ s'écrit

$$m(x_1, x_2) = x_1 \cdot x_2 = \frac{(x_1 + x_2)^2}{4} - \frac{(x_1 - x_2)^2}{4}$$

autrement dit $g_1(x) = x^2/4$, $g_1(x) = -x^2/4$, $\boldsymbol{\omega}_1 = (1, 1)^\intercal$ et $\boldsymbol{\omega}_1 = (1, -1)^\intercal$. Dans la version simple, avec $k = 1$, avec une fonction de perte quadratique, on peut utiliser un développement de Taylor pour approcher $[y_i - g(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i)]^2$, et construire classiquement un algorithme itératif. Si on dispose d'une valeur initiale $\boldsymbol{\omega}_0$, notons que

$$\sum_{i=1}^{n} [y_i - g(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i)]^2 \approx \sum_{i=1}^{n} g'(\boldsymbol{\omega}_0^\intercal \boldsymbol{x}_i)^2 \left[\boldsymbol{\omega}^\intercal \boldsymbol{x}_i + \frac{y_i - g(\boldsymbol{\omega}_0^\intercal \boldsymbol{x}_i)}{g'(\boldsymbol{\omega}_0^\intercal \boldsymbol{x}_i)} - \boldsymbol{\omega}^\intercal \boldsymbol{x}_i \right]^2$$

qui correspondrait à l'approximation dans les modèles linéaires généralisés sur la fonction $g(\cdot)$ qui était la fonction de lien (supposée connue). On reconnaît un problème de moindres carrés pondérés. La difficulté ici est que les fonctions $g_j(\cdot)$ sont inconnues.

3.5 APPLICATIONS

Les données massives ont rendu nécessaire le développement de techniques d'estimation permettant de pallier les limites des modèles paramétriques, jugés trop restrictifs, et des modèles non-paramétriques classiques, dont l'estimation peut être difficile en présence d'un nombre élevé de variables. L'*apprentissage statistique*, ou *apprentissage machine*, propose de nouvelles méthodes d'estimation non-paramétriques, performantes dans un cadre général et en présence d'un grand nombre de variables.[11] Toutefois, l'obtention d'une plus grande flexibilité s'obtient au prix d'un manque d'interprétation qui peut être important.

En pratique, une question importante est de savoir quel est le meilleur modèle ? La réponse à cette question dépend du problème sous-jacent. Si la relation entre les variables est correctement approximée par un modèle linéaire, un modèle paramétrique correctement spécifié devrait être performant. Par contre, si le modèle paramétrique n'est pas correctement spécifié, car la relation est fortement non-linéaire et/ou fait intervenir des effets croisés non-négligeables, alors les méthodes statistiques issues de l'apprentissage automatique devraient être plus performantes.

La bonne spécification d'un modèle de régression est une hypothèse souvent posée, elle est rarement vérifiée et justifiée. Dans les applications qui suivent, nous montrons comment les méthodes statistiques issues de l'apprentissage automatique peuvent être utilisées pour justifier la bonne spécification d'un modèle de régression paramétrique, ou pour détecter une mauvaise spécification. Des applications en classification sont présentées dans un premier temps, sections 3.5.1, 3.5.2 et 3.5.3. D'autres applications sont ensuite présentées dans le contexte de régression classique, sections 3.5.4 et 3.5.5.

3.5.1 *Les ventes de sièges auto pour enfants (classification)*

Nous reprenons ici un exemple utilisé dans James and Tibshirani (2013). Le jeu de données contient les ventes de sièges auto pour enfants dans 400 magasins (Sales), ainsi que plusieurs variables, dont la qualité de présentation en rayonnage (Shelveloc, égal à « mauvais », « moyen », « bon ») et le prix (Price).[12] Une variable dépendante binaire est artificiellement créée, pour qualifier une forte vente ou non (High=« oui » si Sales > 8 et à « non » sinon). Dans cette application, on cherche à évaluer les déterminants d'un bon niveau de vente.

Dans un premier temps, on considère un modèle de régression linéaire latent:

$$y^\star = \gamma + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} + \varepsilon, \qquad \varepsilon \sim G(0,1), \tag{18}$$

11 Entre autres, voir Hastie and Friedman (2009) et James and Tibshirani (2013).

12 C'est le jeu de données Carseats de la bibliothèque ISLR.

où $\boldsymbol{x}$ est composé de k variables explicatives, $\boldsymbol{\beta}$ est un vecteur de k paramètres inconnus et ε est un terme d'erreur *i.i.d.* avec une fonction de répartition G d'espérance nulle et de variance égale à un. La variable dépendante $y^\star$ n'est pas observé, mais seulement y, avec:

$$y = \begin{cases} 1 & \text{si } y^\star > \xi, \\ 0 & \text{si } y^\star \leq \xi. \end{cases} \tag{19}$$

On peut alors exprimer la probabilité d'avoir y égal à 1, comme suit :

$$\mathbb{P}(Y = 1) = G(\beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}) \tag{20}$$

où $\beta_0 = \gamma - \xi$.[13] L'estimation de ce modèle se fait par maximum de vraisemblance en sélectionnant *a priori* une loi paramétrique G. Si on suppose que G est la loi Normale, c'est un modèle probit, si on suppose que G est la loi logistique, c'est un modèle logit. Dans un modèle logit/probit, il y a deux sources potentielles de mauvaise spécification :

(i) la relation linéaire $\beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}$ est mal spécifiée

(ii) la loi paramétrique utilisée G n'est pas la bonne

En cas de mauvaise spécification, de l'une ou l'autre sorte, l'estimation n'est plus valide. Le modèle le plus flexible est le suivant :

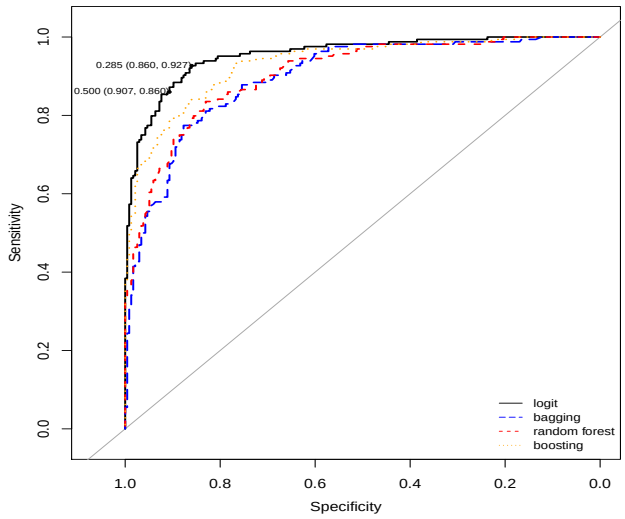
$$\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = G(h(\boldsymbol{x})) \tag{21}$$

où h est une fonction inconnue et G une fonction de répartition inconnue. Les modèles de « *bagging* », de forêt aléatoire et de « *boosting* » permettent d'estimer ce modèle général sans faire de choix *à priori* sur la fonction h et sur la distribution G. L'estimation du modèle logit/probit est néanmoins plus performante si h et G sont correctement spécifiés.

Nous estimons le modèle (20) avec la loi logistique pour G, et le modèle (21) avec les méthodes de « *bagging* », de forêt aléatoire et de « *boosting* ». Nous faisons une analyse de validation croisée par 10 blocs. Les probabilités individuelles des données *out-of-sample*, c'est à-dire de chacun des blocs non-utilisée pour l'estimation, sont utilisées pour évaluer la qualité de la classification.

La figure 55 présente la courbe ROC, ainsi que l'aire sous la courbe (AUC), pour les estimations logit, « *bagging* », forêt aléatoire et « *boosting* ». La courbe ROC est un graphique qui représente simultanément la qualité de la prévision dans les deux classes, pour des valeurs différentes du seuil utilisé pour classer les individus (on parle de « *cutoff* »). Une manière naturelle de classer les individus consiste à les attribuer dans la classe

13 $\mathbb{P}[Y = 1] = \mathbb{P}[Y^\star > \xi] = \mathbb{P}[\gamma + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} + \varepsilon > \xi] = \mathbb{P}[\varepsilon > \xi - \gamma - \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}] = \mathbb{P}[\varepsilon < \gamma - \xi + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}]$. En posant $\gamma - \xi = \beta_0$, on obtient $\mathbb{P}[Y = 1] = G(\beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta})$. En général, on suppose que le terme d'erreur est de variance σ^2, auquel cas les paramètres du modèle (20) deviennent β_0/σ et $\boldsymbol{\beta}/\sigma$, ce qui veut dire que les paramètres du modèle latent (18) ne sont pas identifiables, ils sont estimés à un paramètre d'échelle près.

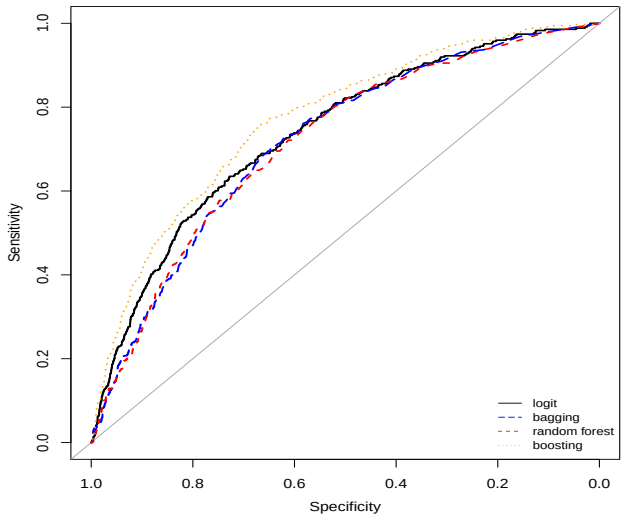


	AUC
logit	0.9544
bagging	0.8973
random forest	0.9050
boosting	0.9313

Figure 55.: Ventes de sièges auto: courbes ROC et aires sous la courbe (AUC).

pour laquelle ils ont la plus grande probabilité estimée. Dans le cas d'une variable binaire, cela revient à prédire la classe d'appartenance pour laquelle la probabilité estimée est supérieure à 0.5. Mais un autre seuil pourrait être utilisé. Par exemple, dans la figure 55, un point de la courbe ROC du modèle logit indique qu'en prenant un seuil égal à 0.5, la réponse « non » est correctement prédite à 90.7% (specificity), et la réponse « oui » à 86% (sensitivity). Un autre point indique qu'en prenant un seuil égal à 0.285, la réponse « non » est correctement prédite à 86% (specificity), et la réponse « oui » à 92.7% (sensitivity). Comme décrit auparavant, un modèle de classification idéal aurait une courbe ROC de la forme Γ. Autrement dit, le meilleur modèle est celui dont la courbe est au-dessus des autres. Un critère souvent utilisé pour sélectionner le meilleur modèle est celui dont l'aire sous la courbe ROC est la plus grande (AUC). L'avantage d'un tel critère est qu'il est simple à comparer et qu'il ne dépend pas du choix du seuil de classification.

Dans notre exemple, la courbe ROC du modèle logit domine les autres courbes, et son aire sous la courbe est la plus grande (AUC=0.9544). Ces résultats indiquent que ce modèle fournit les meilleures prévisions de classification. N'étant dominé par aucun autre modèle, ce constat suggère que le modèle linéaire logit est correctement spécifié et qu'il n'est pas utile d'utiliser un modèle plus général et plus complexe.



	AUC
logit	0.7372
bagging	0.7198
random forest	0.7154
boosting	0.7691

Figure 56.: Achat d'assurance: courbes ROC et aires sous la courbe (AUC).

3.5.2 *L'achat d'une assurance caravane (classification)*

Nous reprenons à nouveau un exemple utilisé dans James and Tibshirani (2013). Le jeu de données contient 85 variables sur les caractéristiques démographiques de 5822 individus.[14] La variable dépendante (Purchase) indique si l'individu a acheté une assurance caravane, c'est une variable binaire, égale à « oui » ou « non ». Dans le jeu de données, seulement 6% des individus ont pris une telle assurance. Les classes sont donc fortement déséquilibrées.

Nous estimons le modèle (20) avec la loi logistique et le modèle (21) avec les méthodes « *bagging* », forêt aléatoire et « *boosting* » (les paramètres de « *tuning* » sont ceux de James and Tibshirani (2013), n.trees=1000 et shrinkage=0.01). Nous faisons une analyse de validation croisée par 10 blocs. Les probabilités individuelles des données *out-of-sample*, c'est à-dire de chacun des blocs non-utilisée pour l'estimation, sont utilisées pour évaluer la qualité de la classification.

La figure 56 présente la courbe ROC, ainsi que l'aire sous la courbe (AUC), pour les estimations logit, « *bagging* », forêt aléatoire et « *boosting* ». La courbe du modèle boosting domine les autres courbes, son aire sous la courbe est la plus grande (AUC=0.7691). Ces résultats indiquent que le boosting fournit les meilleures prévisions de classification. Notons que, comparées à l'exemple précédent, les courbes sont assez éloignées de la forme en coude, ce qui suggère que la classification ne sera pas aussi bonne.

14 C'est le jeu de données Caravan de la bibliothèque ISLR sous R.

	0.5 cutoff		cutoff optimal	
	spécificité	sensitivité	spécificité	sensitivité
logit	0.9967	0.0057	0.7278	0.6351
bagging	0.9779	0.0661	0.6443	0.7069
random forest	0.9892	0.0316	0.6345	0.6954
boosting	0.9987	0.0000	0.6860	0.7385

Table 14.: Achat d'assurance: sensibilité au choix du seuil de classification.

Il faut faire attention aux résultats d'une classification standard, c'est-à-dire avec un seuil de classification égal à 0.5, qui est souvent pris par défaut dans les logiciels (la prédiction de la réponse de l'individu i est « non » si la probabilité estimée qu'il réponde « non » est supérieure à 0.5, sinon c'est « oui »). La partie gauche du tableau 14 présente les taux de classifications correctes avec ce seuil (0.5 cutoff), pour les différentes méthodes. Avec le meilleur modèle et le seuil standard (boosting et seuil à 0.5), les réponses « non » sont correctes à 99.87% (spécificité, specificity) et les réponses « oui » sont toutes fausses (sensitivité, sensitivity). Autrement dit, cela équivaut à utiliser un modèle qui prédit que personne n'achète d'assurance caravane. Sélectionner un tel modèle est absurde pour l'analyste, qui est surtout intéressé par les 6% des individus qui en ont pris une. Ce résultat s'explique par la présence de classes fortement déséquilibrées. En effet, dans notre exemple, en prévoyant que personne n'achète d'assurance, on fait « seulement » 6% d'erreur. Mais ce sont des erreurs qui conduisent à ne rien expliquer.

Plusieurs méthodes peuvent être utiles pour pallier à ce problème, lié aux classes fortement déséquilibrées (pour plus d'informations, voir Kuhn and Johnson (2013), chapitre 16). Une solution simple consiste à utiliser un seuil de classification différent. La courbe ROC présente les résultats en fonction de plusieurs seuils de classification, où la classification parfaite est illustrée par le couple (specificity, sensitivity)=(1,1), c'est à-dire par le coin supérieur gauche dans le graphique. Aussi, on choisit comme seuil de classification optimal celui qui correspond au point de la courbe ROC qui est le plus proche du point (1,1), ou du coin supérieur gauche. La partie droite du tableau 14 présente les taux de classifications correctes avec les seuils optimaux (*optimal cutoff*), pour les différentes méthodes (les seuils optimaux des méthodes logit, « *bagging* », forêt aléatoire et « *boosting* » sont, respectivement, égaux à $0.0655, 0.0365, 0.0395, 0.0596$). Avec le boosting et un seuil optimal, les réponses « non sont correctes à 68.6% (specificity) et les réponses « oui » à 73.85% (sensitivity). L'objet de l'analyse étant de prévoir correctement les individus susceptibles d'acheter une assurance caravane (classe « oui »), et les distinguer suffisamment des autres (classe « non »), le choix du seuil optimal est beaucoup plus performant que le seuil standard 0.5. Notons qu'avec un modèle logit et un seuil optimal, le taux de classifications correctes de la classe « non » est de 72.78%, celui de la classe

« oui » est de 63.51%. Par rapport au boosting, le logit prédit un peu mieux la classe « non », mais nettement moins bien la classe « oui ».

3.5.3 *Les défauts de remboursement de crédits particuliers (classification)*

Considérons la base allemande de crédits particuliers, utilisée dans Nisbet and Miner (2011) et Tufféry (2001), avec 1000 observations, et 19 variables explicatives, dont 12 qualitatives c'est à dire, en les disjonctant (en créant une variable indicatrice pour chaque modalité), 48 variables explicatives potentielles.

Une question récurrente en modélisation est de savoir quelles sont les variables qui mériteraient d'être utilisées. La réponse la plus naturelle pour un économètre pourrait être une méthode de type *stepwise* (parcourir toutes les combinaisons possibles de variables étant a priori un problème trop complexe en grande dimension). La suite des variables dans une approche *forward* est présentée dans la première colonne du tableau 15. Une approche mentionnée avant qui peut être utile est le LASSO, en pénalisant convenablement la norme ℓ_1 du vecteur de paramètres $\boldsymbol{\beta}$. On peut ainsi, séquentiellement, trouver les valeurs du paramètre de pénalisation λ, qui permet d'avoir une variable explicative supplémentaire, non nulle. Ces variables sont présentées dans la dernière colonne. On note que les deux premières variables considérées comme non nulles (pour un λ assez grand) sont les deux premières à ressortir lors d'une procédure *forward*. Enfin, une dernière méthode a été proposée par Breiman (2001b), en utilisant tous les arbres créé lors de la construction d'une forêt aléatoire : l'importance de la variable x_k dans une forêt de T arbres est donnée par:

$$\text{Importance}(x_k) = \frac{1}{T}\sum_{t=1}^{n}\sum_{j\in N_{t,k}} p_t(j)\Delta\mathcal{I}(j)$$

où $N_{t,k}$ désigne l'ensemble des nœuds de l'arbre t utilisant la variable x_k comme variable de séparation, $p_t(j)$ désigne la proportion des observations au nœud j, et $\Delta(j)$ est la variation d'indice au nœud j (entre le nœud précédant, la feuille de gauche et celle de droite). Dans la colonne centrale du tableau 15 sont présentées les variables par ordre d'importance décroissante, lorsque l'indice utilisé est l'indice d'impureté de Gini.

Avec l'approche *stepwise* (détaillée en Annexe A.2) et l'approche LASSO, on reste sur des modèles logistiques linéaires. Dans le cas des forêts aléatoires (et des arbres), des interactions entre variables peuvent être prises en compte, lorsque 2 variables sont présentes. Par exemple la variable residence_since est présente très haut parmi les variables prédictives (troisième variable la plus importante).

Stepwise	AIC	Random Forest	Gini	LASSO
checking_statusA14	1112.1730	checking_statusA14	30.818197	checking_statusA14
credit_amount(4e+03,Inf]	1090.3467	installment_rate	20.786313	credit_amount(4e+03,Inf]
credit_historyA34	1071.8062	residence_since	19.853029	credit_historyA34
installment_rate	1056.3428	duration(15,36]	11.377471	duration(36,Inf]
purposeA41	1044.1580	credit_historyA34	10.966407	credit_historyA31
savingsA65	1033.7521	credit_amount	10.964186	savingsA65
purposeA43	1023.4673	existing_credits	10.482961	housingA152
housingA152	1015.3619	other_payment_plansA143	10.469886	duration(15,36]
other_payment_plansA143	1008.8532	telephoneA192	10.217750	purposeA41
personal_statusA93	1001.6574	age	10.071736	installment_rate
savingsA64	996.0108	savingsA65	9.547362	property_magnitudeA124
other_partiesA103	991.0377	checking_statusA12	9.502445	age(25,Inf]
checking_statusA13	985.9720	housingA152	8.757095	checking_statusA13
checking_statusA12	982.9530	jobA173	8.734460	purposeA43
employmentA74	980.2228	personal_statusA93	8.715932	other_partiesA103
age(25,Inf]	977.9145	property_magnitudeA123	8.634527	employmentA72
purposeA42	975.2365	personal_statusA92	8.438480	savingsA64
duration(15,36]	972.5094	purposeA43	8.362432	employmentA74
duration(36,Inf]	966.7004	employmentA73	8.225416	purposeA46
purposeA49	965.1470	employmentA75	8.089682	personal_statusA93
purposeA410	963.2713	duration(36,Inf]	8.029945	personal_statusA92
credit_historyA31	962.1370	purposeA42	8.025749	savingsA63
purposeA48	961.1567	property_magnitudeA122	7.908813	telephoneA192

Table 15.: Crédit: choix de variables, tri séquentiel, par approche *stepwise*, par fonction d'importance dans une forêt aléatoire et par LASSO.

3.5.4 *Les déterminants des salaires (régression)*

Afin d'expliquer les salaires (individuels) en fonction du niveau d'étude, de l'expérience de la personne, et son genre, il est classique d'utiliser l'équation de salaire de Mincer - décrite dans Mincer (1974) - tel que le rappelle Lemieux (2006):

$$\log(\mathsf{wage}) = \beta_0 + \beta_1\,\mathsf{ed} + \beta_2\,\mathsf{exp} + \beta_3\,\mathsf{exp}^2 + \beta_4\,\mathsf{fe} + \varepsilon \tag{22}$$

où ed est le niveau d'études, ex l'expérience professionnelle et fe une variable indicatrice, égale à 1 si l'individu est une femme et à 0 sinon. D'après la théorie du capital humain, le salaire espéré augmente avec l'expérience, de moins en moins vite, pour atteindre un maximum avant de diminuer. L'introduction du carré de exp permet de prendre en compte une telle relation. La présence de la variable fe permet quand à elle de mesurer une éventuelle discrimination salariale entre les hommes et les femmes.

Le modèle (22) impose une relation linéaire entre le salaire et le niveau d'étude, et une relation quadratique entre le salaire et l'expérience professionnelle. Ces relations peuvent paraître trop restrictives. Plusieurs études montrent notamment que le salaire ne diminue pas après un certain age, et qu'une relation quadratique ou un polynôme de degré plus élevé est plus adapté (comme décrit dans Murphy and Welch (1990) et Bazen and Charni (2015)).

Le modèle (22) impose également que la différence salariale entre les hommes et les femmes est indépendante du niveau d'étude et de l'expérience. Il est trop restrictif si, par exemple, on suspecte que l'écart de salaire moyen entre les hommes et les femmes est faible pour les postes non-qualifiés et fort pour les postes qualifiés, ou faible en début de carrière et fort en fin de carrière (*effets d'intéractions*).

Le modèle le plus flexible est le modèle entièrement non-paramétrique :

$$\log(\mathsf{wage}) = m(\mathsf{ed},\,\mathsf{exp},\,\mathsf{fe}) + \varepsilon \tag{23}$$

où $m(\cdot)$ est une fonction quelconque. Il a l'avantage de pouvoir tenir compte de relations non-linéaires quelconques et d'interactions complexes entre les variables. Mais, sa grande flexibilité se fait au détriment d'une interprétation plus difficile du modèle. En effet, il faudrait un graphique en 4-dimensions pour représenter la fonction m. Une solution consiste à représenter la fonction m en 3 dimensions, en fixant la valeur de l'une des variables, mais la fonction représentée peut être très différente avec une valeur fixée différente.

Nous utilisons les données d'une enquête de l'US Census Bureau daté de mai 1985, issues de l'ouvrage de Berndt (1990) et disponibles sour R.[15] Nous estimons les deux modèles et utilisons une analyse de validation croisées par 10 blocs pour sélectionner la meilleure approche. Le modèle paramétrique (22) est estimé par Moindres Carrés Ordinaires (OLS). Le modèle entièrement non-paramétrique (23) est estimé par la méthode

15 C'est le jeu de données CPS1985 de la bibliothèque AER.

	Modèle (22)	Modèle (23)			
$\widehat{\mathcal{R}}^{10-\mathsf{CV}}$	OLS	Splines	Bagging	R.Forest	Boosting
out-of-sample	0.2006	0.2004	0.2762	0.2160	0.2173

Table 16.: Salaire: analyse de validation croisée par blocs ($K = 10$) : performances de l'estimation des modèles linéaire (22) et entièrement non-paramétrique (23).

des *splines*, car il en comprend peu de variables, ainsi que par les méthodes *bagging*, *random forest* et *boosting*.

Le tableau 16 présente les résultats de la validation croisée en 10 blocs (*10-fold cross-validation*). Le meilleur modèle est celui qui minimise le critère $\widehat{\mathcal{R}}^{10-\mathsf{CV}}$. Les résultats montrent que le modèle (22) est au moins aussi performant que le modèle (23), ce qui suggère que le modèle paramétrique (22) est correctement spécifié.

3.5.5 *Les déterminants des prix des logements à Boston (régression)*

Nous reprenons ici l'un des exemples utilisé dans James and Tibshirani (2013), dont les données sont disponibles sous R. Le jeu de données contient les valeurs médianes des prix des maisons (medv) dans $n = 506$ quartiers autour de Boston, ainsi que 13 autres variables, dont le nombre moyen de pièces par maison (rm), l'age moyen des maisons (age) et le pourcentage de ménages dont la catégorie socio-professionnelle est peu élevée (lstat).[16]

Considérons le modèle de régression linéaire suivant :

$$\mathsf{medv} = \alpha + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta} + \varepsilon \tag{24}$$

où $\boldsymbol{x} = [\mathsf{chas},\mathsf{nox},\mathsf{age},\mathsf{tax},\mathsf{indus},\mathsf{rad},\mathsf{dis},\mathsf{lstat},\mathsf{crim},\mathsf{black},\mathsf{rm},\mathsf{zn},\mathsf{ptratio}]$ est un vecteur en dimension 13 et $\boldsymbol{\beta}$ est un vecteur de 13 paramètres. Ce modèle spécifie une relation linéaire entre la valeur des maisons et chacune des variable explicatives.

Le modèle le plus flexible est le modèle entièrement non-paramétrique :

$$\mathsf{medv} = m(\boldsymbol{x}) + \varepsilon. \tag{25}$$

L'estimation de ce modèle avec les méthodes du noyau ou les splines peut être problématique, car le nombre de variables est relativement élevé (il y a ici 13 variables), ou au moins trop élevé pour envisager estimer une surface en dimension 13. Nous estimons les deux modèles et utilisons une analyse de validation croisée par 10-blocs pour sélectionner la meilleure approche. Le modèle paramétrique (24) est estimé par Moindres Carrés

16 C'est le jeu de données `Boston` de la librairie `MASS`. Pour une description complète des données, voir: https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/Boston.html.

	Modèle (24)	Modèle (25)		
$\widehat{\mathcal{R}}^{10-\mathsf{CV}}$	OLS	Bagging	R.Forest	Boosting
in-sample	21.782	1.867	1.849	7.012
out-of-sample	24.082	9.590	9.407	11.789

Table 17.: Prix des logements à Boston: analyse de validation croisée par blocs ($K = 10$): performances de l'estimation des modèles linéaire (24) et entièrement non-paramétrique (25).

Ordinaires (OLS) et le modèle entièrement non-paramétrique (25) est estimé par trois méthodes différentes: « *bagging* », forêt aléatoire et « *boosting* » (nous utilisons ici les valeurs par défaut utilisées dans James and Tibshirani (2013), pp. 328-331).

Le tableau 17 présente les résultats de la validation croisée en 10 blocs (*10-fold cross-validation*). La première ligne (*in-sample*) présente la qualité de l'ajustement des modèles en utilisant seulement les données d'apprentissage, c'est-à-dire celles qui ont servi à estimer le modèle, pour calculer le MSE. La deuxième ligne (*out-of-sample*) présente la qualité de l'ajustement en utilisant d'autres données que celles ayant servies à estimer le modèle, pour calculer l'erreur quadratique. à partir des résultats in-sample, les méthodes de « *bagging* » et de forêt aléatoire paraissent incroyablement plus performantes que l'estimation OLS du modèle linéaire (24), le critère $\widehat{\mathcal{R}}^{10-\mathsf{CV}}$ passant de 21.782 à 1.867 et 1.849. Les résultats *out-of-sample* vont dans le même sens, mais la différence est moins importante, le critère $\widehat{\mathcal{R}}^{10-\mathsf{CV}}$ passant de 24.082 à 9.59 et 9.407. Ces résultats illustrent un phénomène classique des méthodes non-linéaires, comme le « *bagging* » et la forêt aléatoire, qui peuvent être très performantes pour prédire les données utilisées pour l'estimation, mais moins performantes pour prédire des données hors-échantillon. C'est pourquoi la sélection de la meilleure estimation est habituellement basée sur une analyse *out-of-sample*, telle que présentée dans la deuxième ligne.

La différence entre l'estimation du modèle linéaire (24) et du modèle entièrement non-paramétrique (25) est importante (24.082 vs 9.590, 9.407 et 11.789). Un tel écart suggère que le modèle linéaire est mal spécifié, et que des relations non-linéaire et/ou des effets d'interactions sont présentes dans la relation entre le prix des logements, medv, et les variables explicatives $\boldsymbol{x}$. Ce résultat nous conduit à chercher une meilleure spécification paramétrique.

à partir du modèle paramétrique (24), et afin de prendre en compte d'éventuelles non-linéarités, le modèle additif généralisé (GAM) suivant peut être considéré :

$$\mathsf{medv} = m_1(x_1) + m_2(x_2) + \cdots + m_{13}(x_{13}) + \varepsilon, \tag{26}$$

où $m_1, m_2, \ldots m_{13}$ sont des fonctions inconnues. L'avantage de ce modèle est qu'il permet de considérer n'importe quelle relation non-linéaire entre la variable dépendante et

	%IncMSE	IncNodePurity
rm	61.35	18345.41
lstat	36.20	15618.22
dis	29.37	2601.72
nox	24.91	1034.71
age	17.86	554.50
ptratio	17.43	626.58
tax	16.60	611.37
crim	16.26	1701.73
indus	9.45	237.35
black	8.72	457.58
rad	4.53	166.72
zn	3.10	35.73
chas	0.87	39.05

Table 18.: Prix des logements à Boston: mesures de l'importance de chacune des variables dans l'estimation de forêt aléatoire du modèle (25), en considérant tout l'échantillon.

chacune des variables explicatives. De plus, il ne souffre pas du problème du fléau de la dimension, car chacune des fonction est de dimension 1, et il est facilement interprétable. Toutefois, il ne prend pas en compte d'éventuels effets d'intéractions.

L'estimation du modèle additif généralisé (26) par la méthode des *splines*, dans le cadre d'une analyse de validation croisée par 10-blocs, donne une valeur $\widehat{\mathcal{R}}^{10-\mathsf{CV}} = 13.643$. Par rapport au modèle paramétrique (24), il y a un gain important (13.643 vs. 24.082). Mais la différence avec le modèle entièrement non-paramétrique (25) reste conséquente (13.643 vs 9.590, 9.407, 11.789). Une telle différence suggère que la prise en compte de relations individuelles pouvant être fortement non-linéaires n'est pas suffisante, et que des effets d'intéractions entre les variables sont présents. Nous pourrions inclure dans le modèle les variables d'intéractions les plus simples entre toutes les paires de variables $(x_i \times x_j)$, mais cela impliquerait de rajouter un très grand nombre de variables au modèle initial (78 dans notre cas), qui ne serait pas sans conséquence sur la qualité de l'estimation du modèle. Quoi qu'il en soit, nous pouvons dire pour le moment que le modèle linéaire est mal spécifié et qu'il existe des effets d'intéractions pouvant être forts dans la relation entre medv et X, l'identification de tels effets restant délicat.

Afin d'aller plus loin, les outils développés en apprentissage statistique peuvent être à nouveau d'un grand recours. Par exemple, l'estimation de forêt aléatoire s'accompagne de mesures de l'importance de chacune des variables dans l'estimation du modèle (décrit dans la section précédente). Le tableau 18 présente ces mesures dans le cadre du modèle (25), estimé sur l'échantillon complet. Les résultats suggèrent que les variables rm et lstat sont les variables les plus importantes pour expliquer les variations des prix des

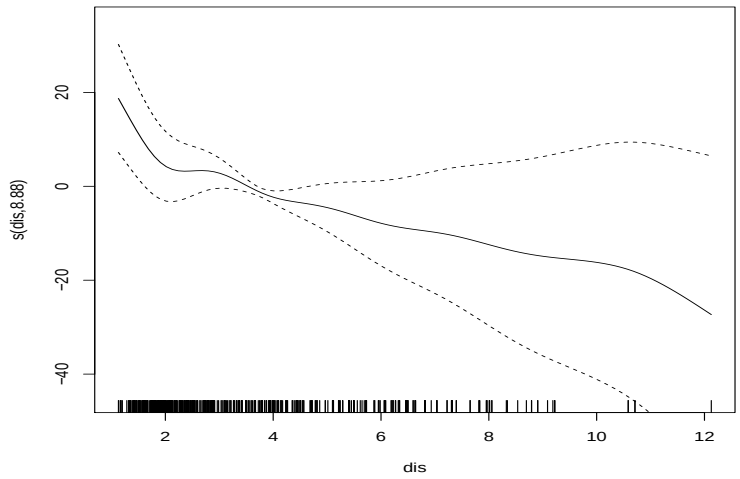


Figure 57.: Estimation de la relation $m_7(x_7)$ dans le modèle additif généralisé (27), où $x_7 = \mathsf{dis}$.

logements medv. Ce constat nous conduit à enrichir la relation initiale, en rajoutant les d'interactions liées à ces deux variables seulement, qui sont les plus importantes.

Nous estimons le modèle additif généralisé incluant les variables d'intéractions, sur l'échantillon complet:

$$\mathsf{medv} = m_1(x_1) + \cdots + m_{13}(x_{13}) + (\mathsf{rm}{:}x)\,\gamma + (\mathsf{lstat}{:}x)\,\delta + \varepsilon, \tag{27}$$

où $(\mathsf{rm}{:}x)$ représente les variables d'intéractions de rm avec toutes les autres variables de x et $(\mathsf{lstat}{:}x)$ représente les variables d'intéractions de lstat avec toutes les autres variables de x.[17] L'analyse des résultats de cette estimation suggère que les fonctions $\hat{m}_i$ sont linéaires pour toutes les variables, sauf pour la variable dis, dont la relation estimée est présentée dans la figure 57. Cette variable mesure la distance moyenne à cinq centres d'emplois de la région. L'effet semble diminuer plus rapidement avec la distance, lorsque celle-ci n'est pas très élevée. Au delà d'une certaine distance (au delà de 2, en log), l'effet est réduit, il continue à diminuer mais plus doucement. Cette relation non-linéaire peut être approchée par une régression linéaire par morceaux en considérant un nœud.

Finalement, l'analyse précédente nous conduit à considérer le modèle linéaire suivant:

$$\mathsf{medv} = \alpha + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta} + (\mathsf{dis} - 2)_+\,\theta + (\mathsf{rm}{:}x)\,\gamma + (\mathsf{lstat}{:}x)\,\delta + \varepsilon \tag{28}$$

17 On a $(\mathsf{rm}{:}x)$ = [rm×chas, rm×nox, rm×age, rm×tax, rm×indus, rm×rad, rm×dis, rm×lstat, rm×crim, rm×black, rm×zn, rm×ptratio] et $(\mathsf{lstat}{:}x)$ = [lstat×chas, lstat×nox, lstat×age, lstat×tax, lstat×indus, lstat×rad, lstat×dis, lstat×crim, lstat×black, lstat×zn, lstat×ptratio].

$\widehat{\mathcal{R}}^{10-\mathsf{CV}}$	Modèle (24) OLS	Modèle (26) Splines	Modèle (28) OLS
out-of-sample	24.082	13.643	11.759

Table 19.: Prix des logements à Boston: analyse de validation croisée par blocs ($K = 10$) : performances de l'estimation du modèle linéaire (24) et du modèle linéaire (28) incluant les effets d'interactions et une non-linéarité par morceaux.

où $(\mathsf{dis} - 2)_+$ est égal à la valeur de son argument si ce dernier est positif, et à 0 sinon. Par rapport au modèle linéaire initial, ce modèle inclut une relation linéaire par morceaux avec la variable dis, ainsi que des effets d'interactions entre rm, lstat et chacune des autres variables de $\boldsymbol{x}$.

Le tableau 19 présente les résultats de la validation croisée en 10 blocs (*10-fold cross-validation*) de l'estimation des modèles paramétriques (24) et (28), estimés par Moindres Carrés Ordinaires (OLS), et du modèle additif généralisé (26) estimé par les splines. Il montre que l'ajout des variables d'intéractions et de la relation linéaire par morceaux dans le modèle (28) donne des résultat beaucoup plus performants que le modèle initial (24): le critère $\widehat{\mathcal{R}}^{10-\mathsf{CV}}$ est divisé par plus de deux, il passe de 24.082 à 11.759. En comparant ces résultats avec ceux du tableau 17, on constate également que le modèle paramétrique (28), estimé par OLS, est aussi performant que le modèle général (25) estimé par *boosting* ($\widehat{\mathcal{R}}^{10-\mathsf{CV}} = 11.789$). La différence avec les méthodes *bagging* et forêt aléatoire n'est quant à elle pas très importante ($\widehat{\mathcal{R}}^{10-\mathsf{CV}} = 9.59, 9.407$)

Finalement, les méthodes « *bagging* », forêt aléatoire et « *boosting* » ont permis de mettre en évidence une mauvaise spécification du modèle paramétrique initial, puis de trouver un modèle paramétrique beaucoup plus performant, en prenant compte des effets de non-linéarités et d'interactions appropriées.

3.6 VERS UN NOUVEAU REGARD SUR LES OUTILS TRADITIONNELS

Si les « deux cultures » (ou les deux communautés) de l'économétrie et de l'apprentissage automatique se sont développées en parallèle, le nombre de passerelles entre les deux ne cesse d'augmenter. Alors que Varian (2014) présentait les apports importants de l'économétrie à la communauté de l'apprentissage automatique, nous avons tenté ici de présenter des concepts et des outils développés au fil du temps par ces derniers, qui pourraient être utiles aux économètres, dans un contexte d'explosion du volume de données. Si nous avons commencé par opposer ces deux mondes, c'est aussi pour mieux comprendre leurs forces et leurs faiblesses. Les fondements probabilistes de l'économétrie sont incontestablement sa force, avec non seulement une interprétabilité des modèles, mais aussi une quantification de l'incertitude. Néanmoins, nous l'avons vu à plusieurs

reprises sur des données réelles, les performances prédictives des modèles d'apprentissage automatique sont intéressantes, car elles permettent de mettre en avant une mauvaise spécification d'un modèle économétrique. De la même manière que les techniques non-paramétriques permettent d'avoir un point de référence pour juger de la pertinence d'un modèle paramétrique, les outils d'apprentissage automatique permettent d'améliorer un modèle économétrique, en détectant un effet non-linéaire ou un effet croisé oublié.

Une illustration des interactions possibles entre les deux communautés se trouve par exemple dans Belloni and Hansen (2010)Belloni and Hansen (2012), dans un contexte de choix d'instrument dans une régression. Reprenant les données de Angrist and Krueger (1991) sur un problème de réussite scolaire, ils montrent comment mettre en œuvre efficacement les techniques d'économétrie instrumentale quand on peut choisir parmi 1530 instruments disponibles (problème qui deviendra récurrent avec l'augmentation du volume de données). Ainsi, comme nous l'avons vu tout au long de ce chapitre, même si les approches peuvent être fondamentalement différentes dans les deux communautés, bon nombre d'outils développés par la communauté de l'apprentissage automatique méritent d'être utilisés par les économètres et les actuaires, utilisateurs des mêmes modèles.

4

LE DEEP LEARNING AU SERVICE DE L'ANALYSE AUTOMATIQUE DE DOCUMENTS

Comme le soulève le chapitre 3, l'existence des modèles de réseaux de neurones date de plusieurs dizaines d'années. Partant des premiers réseaux de neurones linéaires dans les années soixante [Widrow and Hoff 1960] jusqu'aux architectures plus complexes et multi-couches comme celles des réseaux convolutionnels [Lecun and Bengio 1995] ou encore récurrents [Pineda 1987] -rendues possibles grâces aux avancées sur les méthodes d'optimisation et notamment de la *backpropagation* [Rumelhart et al. 1986]-, ces modèles constituent aujourd'hui une référence dans de nombreux domaines impliquant le plus souvent des données non structurées (textes, images, sons, etc.). Plus communément désignés par leur appellation anglaise *Deep Learning* de part le nombre de paramètres et de couches utilisées, ces algorithmes ont connu un grand essor grâce à la démocratisation des techniques de parallélisation informatique [Owens et al. 2008] et des librairies *open source* permettant d'utiliser facilement les outils de construction de modèles de *deep learning*. Afin d'étudier la pertinence de ces modèles dans l'analyse automatique de document en assurance, nous avons donc tenté de comprendre les méthodes d'analyse de document et tenté notre propre implémentation.

Plus particulièrement, dans le cadre de la thèse CIFRE avec Milliman, nous souhaitions construire une nouvelle offre permettant d'aider les assureurs à exploiter automatiquement les documents textuels qu'ils possèdent. En effet, les compagnies disposent d'un grand nombre de supports contenant de l'information non-structurée comme par exemple des conditions générales, des rapports d'expertise, des contrats d'assurance, des pièces d'identité ou encore des rapports de solvabilité. Ces données sont pour la plupart encore traitées manuellement et simplement stockées numériquement.

Ce chapitre illustre ainsi les travaux menés et les résultats obtenus par ces recherches. Ils ont notamment pu être présentés lors du séminaire "Data Science" organisé par Marie Kratz, Olga Klopp et Salma Jamal[1]. La fin de ce chapitre introduit également les différentes techniques et méthodes qui ont pu être utilisées afin d'analyser l'information textuelle extraite. Ces travaux feront l'objet d'une future publication.

1 http://crear.essec.edu/working-group-on-risk/past-meetings

4.1 LES PROCESSUS D'EXTRACTION D'INFORMATION DE DONNÉES NON STRUCTURÉES

4.1.1 *L'utilisation des documents par les compagnies d'assurance*

L'assurance moderne telle qu'on la connaît de nos jours est une industrie datant de plus de six siècles. La première compagnie d'assurance maritime fut en effet fondée à Gênes en 1424 bien que les premiers contrats d'assurance furent bien antérieurs à cette date [Legardeur 2007]. Depuis son origine, au fondement de l'assurance réside l'existence d'un contrat entre plusieurs parties. L'archivage de ce document est donc primordial. Le plus souvent sous format papier, ce contrat est encore aujourd'hui archivé de manière numérique -le plus souvent à l'aide d'un scanner. De même de nombreuses pièces associées au dossier de l'assuré et de ses sinistres sont également numérisés : conditions générales, constat à l'amiable, pièces justificatives, etc. Si ces documents sont facilement accessibles pour tout humain autorisé à accéder au dossier de l'assuré, les informations -dans leur forme brute- qu'ils contiennent sont cependant difficilement exploitables pour un algorithme. En effet, les systèmes de gestion des dossiers clients possèdent généralement peu de vues d'ensemble des documents. Autrement dit, afin d'être en mesure d'exploiter tous les contrats par exemple et d'effectuer une analyse statistique, il faudrait ouvrir successivement un à un chaque dossier et extraire les pièces afin de les "déplacer" dans un répertoire commun [2]. D'autre part, ces documents contiennent le plus souvent du texte qui est enregistré sous forme d'image. La donnée textuelle est alors lisible mais pas directement disponible dans le document numérique autrement que sous forme de suites de pixels qu'il convient de convertir en caractères.

Si le format numérique permet à tout moment à la compagnie d'assurance de retrouver les modalités liées à ses engagements envers un tiers, il présente néanmoins des limites. Par exemple, afin de vérifier l'authenticité d'un sinistre, la collecte et l'analyse des nombreuses pièces associées au dossier peut prendre plus d'une dizaine de jours afin d'assurer la conformité mais aussi l'absence d'anomalie. Or, de nos jours, la digitalisation et la diffusion des nouvelles technologies proposent aux individus de plus en plus de services rapides[3]. Cette rapidité des services incite les assurances à indemniser également dans des délais raisonnables afin de maintenir une bonne réputation. Ces délais sont alors incompatibles par exemple avec la mise en place d'une suspicion de fraude compte tenu de l'état de disponibilité des informations et de leur format de stockage.

2 Cette opération est normalement réalisée via un système de gestion de bases de données. En pratique, de nombreuses mutuelles ou compagnies d'assurance ont des systèmes où les documents ne sont pas centralisés, ni même parfois indexés.

3 Par exemple dans le domaine des services d'assurance et financiers, limonade.com, alan.com, luko.com, N26.com etc proposent des souscriptions et paiements des sinistres en des temps inférieurs à 48h.

Être en mesure d'analyser automatiquement un document numérique devient alors un enjeu tant au niveau de l'extraction de l'information qu'il contient que dans la fluidification et l'optimisation des processus de souscription ou d'indemnisation. L'automatisation permet en outre de compléter les actions des assureurs et de minimiser le risque opérationnel dans des analyses systématiques. Selon l'ancienneté des documents numérisés, plusieurs types de pièces peuvent être amenées à être analysées. Pour des documents datant d'avant les années 1970, il n'est pas rare de posséder des documents numériques tels que des actes de naissance ou de décès écrits à la main (que nous désignerons comme "manuscrits" par la suite) comme l'illustre l'image 58.

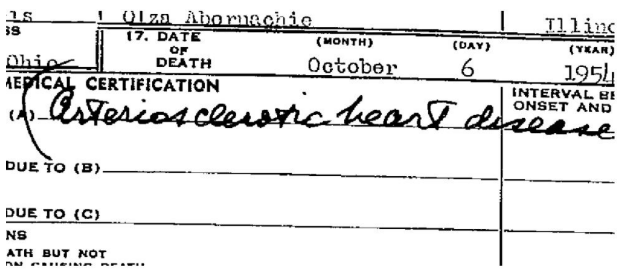


Figure 58.: Exemple de document scanné contenant de l'écriture "manuscrite"

Même pour des documents plus récents datant des dernières décennies, le contenu n'est parfois pas directement accessible (i.e. une recherche d'un paragraphe ou encore une information spécifique n'est pas disponible sans une lecture par un être humain). C'est le cas, des documents ayant été générés via un logiciel ou une machine mais qui ont par la suite été stockés par numérisation d'un exemplaire papier. Ainsi, bien que ces derniers ont une écriture normalisée i.e. non curviligne (que nous appellerons par la suite "tapuscrite"), ils peuvent également être considérés comme une image où le texte ne peut s'extraire directement comme le montre l'image 59.

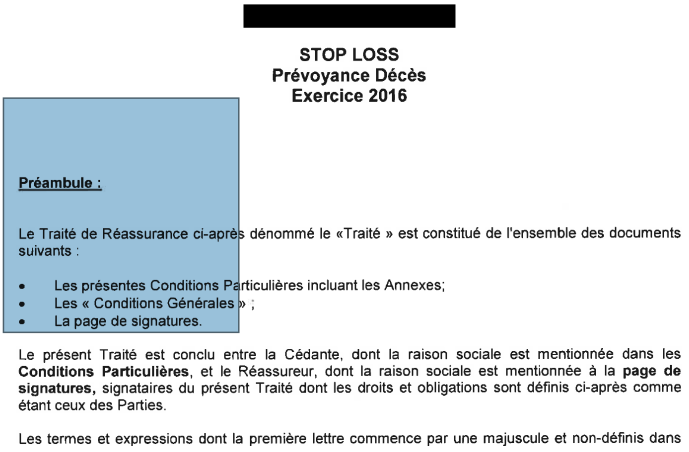


Figure 59.: Exemple de document numérisé, sans accès direct au texte

Dans les deux situations décrites, il est nécessaire de convertir l'objet numérique en informations textuelles qui pourront ensuite être analysées par divers algorithmes que nous décrivons plus loin dans ce chapitre. Nous nous sommes donc intéressés à la Reconnaissance Optique de Caractères dont les premières recherches remontent au début des années 1950 [Cheriet et al. 2007]. On utilisera par la suite le terme anglais OCR : *Optical Charater Recognition*[4] pour faire référence à ces techniques. En d'autres termes, nous avons étudié la conversion d'une image en un texte (qu'il contient potentiellement). Au fondement de cette méthode se pose différentes problématiques parmi lesquelles se trouve la détection des lignes d'écriture et des mots. Une fois détectés, la reconnaissance de ces mots et enfin l'interprétation et l'exploitation de ces derniers nécessite la mise en place d'autres algorithmes que nous avons étudiés. Ainsi, après avoir détaillé les méthodes de détection des lignes de texte ou de mots dans une image, nous exposons l'algorithme de *deep learning* que nous avons calibré afin d'être en mesure de reconnaître des mots et plus particulièrement ceux issus d'écritures "manuscrites".

4.1.2 *L'extraction du texte*

L'utilisation des méthodes d'OCR s'avère nécessaire dès lors qu'un document est numérisé et que l'on souhaite extraire le texte qu'il contient. Formellement, nous pouvons considérer une image I comme un vecteur d'une espace vectoriel. Plus particulièrement par le suite, on considérera I comme étant un tenseur, i.e. un ensemble de $k \in \mathbb{N}^*$ matrices de $\mathcal{M}_{m,n}(\mathbb{R})$. Où $m \times n \in \mathbb{N}^*$ représente le nombre de pixels de l'image. On notera par la suite $\mathcal{I}_k$, l'espace des images où k désigne le nombre de calques composant l'image. Ainsi dans la représentation couramment utilisée RGB (*Red Green Blue)*, une image est

4 Dans la littérature, la première appartition de ce terme remonte à 1971 où il désigne alors la méthode brevetée US3723970A utilisée par le scanner pour convertir une image analogique en numérique.

constituée de 3 matrices (soit 3 calques) dont chacune se décompose en coordonnées indiquant la présence ou non d'un pixel bleu, vert ou rouge. En noir et blanc, une image peut donc simplement être représentée par une unique matrice où chaque coordonnée indique la présence d'un pixel noir comme l'illustre la figure 60.

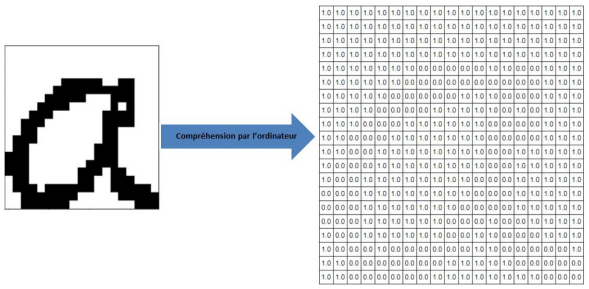


Figure 60.: Représentation matricielle d'une image

Extraire le texte que contient un document revient ainsi à déterminer une fonction $F : \mathcal{I}_k \to \mathcal{C}^{\mathbb{N}}$ c'est-à-dire la transformation qui permet de passer d'une image à une suite de caractères dont l'ensemble est noté $\mathcal{C}$. En première approche nous serions tentés d'utiliser l'apprentissage statistique (cf les chapitres précédents) afin d'approcher cette fonction F de manière supervisé. Or cela nécessite d'observer un échantillon de $n \in \mathbb{N}^*$ couple de données $(I_{k,i}, C_i)$ d'images et des suites de caractères associées qu'elles contiennent (avec $i \in [[1, n]]$ désigne l'indice de l'observation). Il est cependant difficile de collecter de telles données car il faudrait en effet manuellement construire les suites de caractères C_i à partir d'une lecture par un humain d'une image $I_{k,i}$ or c'est précisément ce que l'on cherche à automatiser ici. Afin d'éviter la labellisation manuelle, plusieurs alternatives se sont présentées :

- Utiliser un algorithme d'OCR pré-calibré
- Rechercher une base de données labellisée libre d'accès et calibrer notre propre algorithme
- Adopter une approche non-supervisée d'extraction de texte

Selon nos recherches de littérature, l'approche non-supervisée n'a aujourd'hui jamais été abordée ou du moins n'a donné que des résultats peu concluants. Il existe néanmoins des approches non-supervisées afin d'améliorer les résultats des méthodes supervisées [Rui and A. Smith 2018]. Nous n'explorons pas cette approche dans ce chapitre.

4.1.3 *Etat des lieux des modules existants*

Si les premières méthodes d'OCR se focalisent sur le passage d'une image digitale en numérique [Cheriet et al. 2007], les chercheurs se sont très rapidement intéressés à la numérisation du texte que ces documents contiennent [White and Rohrer 1983]. Ainsi il existe aujourd'hui de nombreux outils proposant l'extraction de texte d'une image. Parmi ces méthodes, nous nous sommes intéressés à l'algorithme *Tesseract* [Smith 2007] et ses performances sur les cas d'usage que nous avons identifiés en introduction de cette section. Par la suite, afin de rationaliser la notion de performance, nous utiliserons la notion de **précision** liée à une fonction d'extraction que nous définissons de la manière suivante :

Définition 4.1. Soit $(I_k, C) \in \mathcal{I}_k \times \mathcal{C}$, une image et la séquence de texte qu'il contient associée. Soit F une fonction d'extraction d'OCR. On définit la **précision** notée *acc* de la fonction F sur (I_k, C) comme

$$acc((I_k, C)) = \frac{1}{|C|} \sum_{(\hat{c},c)\in F(I_k),C} \mathbb{1}(\hat{c}, c)$$

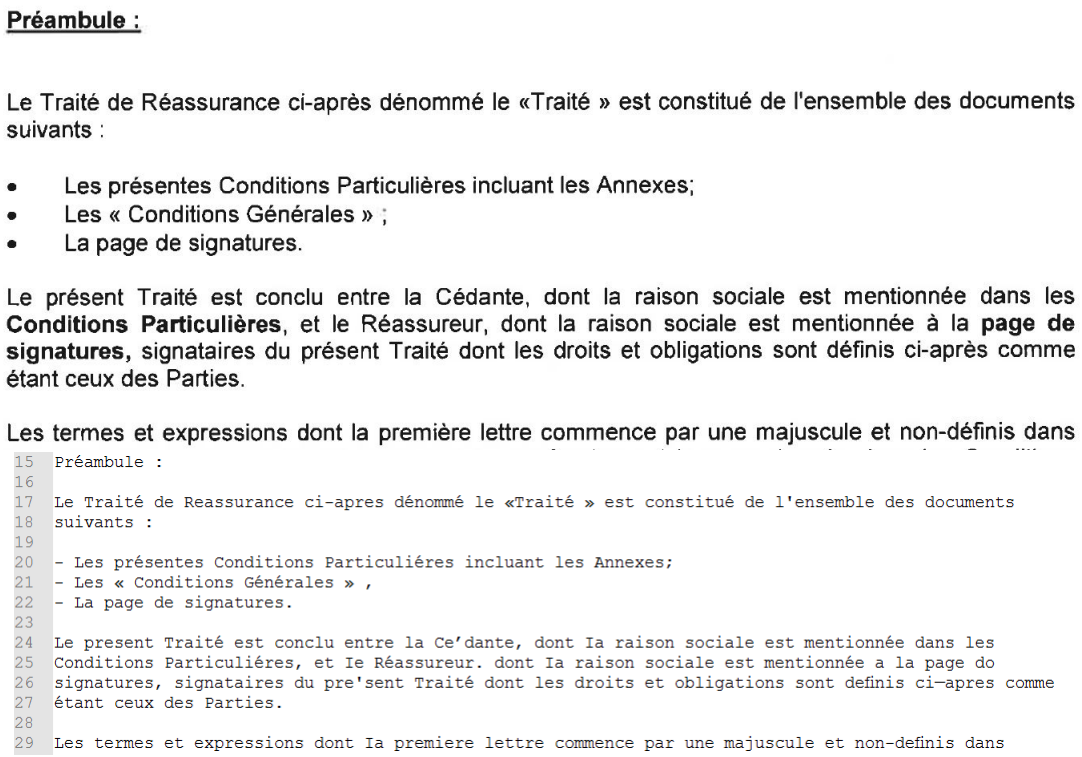
où $|C|$ désigne le nombre d'éléments non nuls de la suite de caractères C.

Autrement dit, la précision évalue le pourcentage de caractères correctement détectés par la fonction d'extraction. L'algorithme *Tesseract* permet aujourd'hui d'obtenir des précisions très intéressantes atteignant en moyenne une précision proche de 100%. Cependant cette précision n'est atteinte que dans certaines conditions bien particulières :

- Le document doit être "tapuscrit", i.e. l'écriture qu'il contient doit être normée et régulière. Cela exclut donc l'écriture curviligne
- Le document digital doit être de "bonne" qualité, c'est-à-dire contenir peu de bruit du par exemple au vieillissement du document (tâches d'encre, contraste, etc.)
- Le document doit contenir des fonds homogènes de couleur unie (par exemple écriture noire sur fond blanc)

L'image 61 page 183 illustre les résultats obtenus par l'algorithme sur un spécimen de traité de réassurance. On constate ainsi qu'à la détection de quelques caractères très spécifiques près, le texte est correctement restitué sous un format numérique qui est directement exploitable par des algorithme d'analyse sémantique que nous détaillons dans le dernière section de ce chapitre.

L'extraction de texte sur des documents détériorés ou très bruités n'est pas détaillée dans ce chapitre. En effet, ces derniers sont de plus en plus minoritaires dans les documents à disposition des assureurs (en partie dû à la modernisation des procédures de



Préambule :

Le Traité de Réassurance ci-après dénommé le «Traité » est constitué de l'ensemble des documents suivants :

- Les présentes Conditions Particulières incluant les Annexes;
- Les « Conditions Générales » ;
- La page de signatures.

Le présent Traité est conclu entre la Cédante, dont la raison sociale est mentionnée dans les **Conditions Particulières**, et le Réassureur, dont la raison sociale est mentionnée à la **page de signatures,** signataires du présent Traité dont les droits et obligations sont définis ci-après comme étant ceux des Parties.

Les termes et expressions dont la première lettre commence par une majuscule et non-définis dans

```
Préambule :

Le Traité de Reassurance ci-apres dénommé le «Traité » est constitué de l'ensemble des documents
suivants :

- Les présentes Conditions Particuliéres incluant les Annexes;
- Les « Conditions Générales » ,
- La page de signatures.

Le present Traité est conclu entre la Ce'dante, dont Ia raison sociale est mentionnée dans les
Conditions Particuliéres, et Ie Réassureur. dont Ia raison sociale est mentionnée a la page do
signatures, signataires du pre'sent Traité dont les droits et obligations sont definis ci—apres comme
étant ceux des Parties.

Les termes et expressions dont Ia premiere lettre commence par une majuscule et non-definis dans
```

Figure 61.: Résultat de l'application de *Tesseract* (en bas) sur un document numérisé de "bonne qualité" (en haut)

souscription et de suivi). Il existe par ailleurs des méthodes permettant d'améliorer les performances des algorithmes comme *Tesseract* notamment en dé-bruitant les images numérisées [Patvardhan et al. 2012]. En l'état cela nécessite de venir modifier certaines couches des outils comme *Tesseract*.

Néanmoins, nous nous sommes intéressés à la question de l'extraction de textes présents sous forme d'écriture "manuscrite" (i.e. curviligne). En effet, les outils libres d'accès n'obtenant pas de bonnes précisions sur l'écriture "manuscrite" (précision inférieure à 10% sur les documents testés), nous avons travaillé à l'élaboration de notre propre algorithme en s'appuyant notamment sur les méthodes de *deep learning*. Nous détaillons ces travaux dans la section suivante.

4.1.4 *Etat des lieux des bases de données disponibles*

Afin d'être en mesure de calibrer notre propre algorithme sans labelliser manuellement l'échantillon d'images[5] sur lesquelles nous souhaitions extraire le texte, il nous a fallu trouver des bases de données. Nos recherches nous ont mené à la découverte de deux sources de données : CVL [Kleber et al. 2013] et RIMES [Grosicki and El Abed 2009]. Cumulées, ces deux bases de données contiennent ainsi un ensemble de plus de 100 000 mots écrits par un panel de plus de 1 500 personnes. Les images représentent alors la retranscription de textes en langue latine (anglais et français). Par ailleurs, certaines images blanches ont pu être également introduites afin de renforcer l'apprentissage des zones vides ou des espaces.

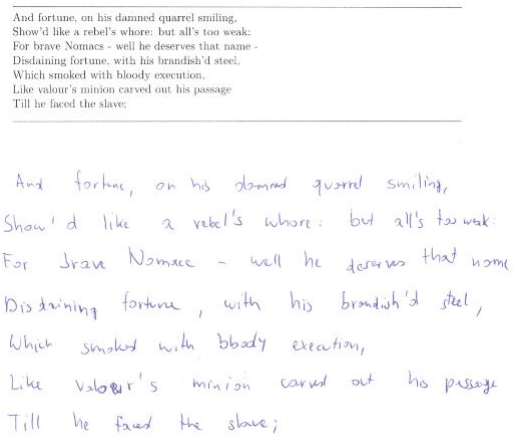


Figure 62.: Aperçu des données de la base CVL

La littérature sur l'implémentation d'un algorithme de *deep learning* à des fins d'OCR [Graves 2013] nous a également mené à des modèles génératifs permettant de simuler des images contenant de l'écriture curviligne[6]. Ces sources de données seront utilisées dans l'implémentation de notre algorithme.

4.2 L'ÉLABORATION DE NOTRE PROPRE ALGORITHME D'OCR

4.2.1 *Description de la démarche*

Avant d'établir notre modèle, on rappelle que les images contenant du bruit ou des fonds non unis peuvent induire en erreur les algorithmes d'OCR [Patvardhan et al. 2012]. Il est ainsi important de posséder des images de "bonne" qualités afin d'obtenir une extraction

5 Des exemplaires de constats automobiles à l'amiable, des actes de décès et de naissance ou des formulaires.
6 https://github.com/Grzego/handwriting-generation

fiable. Nous avons donc supposé dans un premier temps que nos images étaient sur fond unique et qu'elles étaient non bruitées.

En effet, connaissant éventuellement les formulaires ou les documents d'origine avant leur complétion (par exemple formulaire type d'un constat automobile ou un questionnaire), il est raisonnable de déterminer les zones d'intérêt par simple découpage numérique prédéfini de l'image contenant le texte. Cela suppose néanmoins que l'individu respecte scrupuleusement les zones d'écriture prévues à cet effet. Nous considérons donc que nos échantillons d'image ne contiennent que du texte. Nous reviendrons par la suite sur ces différentes contraintes de qualité de l'image.

On rappelle par ailleurs, que les modules existants comme *Tesseract* étant performants sur des documents "tapuscrits" (i.e. contenant de l'écriture régulière et normée), nous nous intéressons dans ce qui suit, essentiellement aux algorithmes d'extraction d'écriture manuscrite (i.e. curviligne). Pour ce faire, nous avons dans un premier temps approfondi les différentes étapes qui composent les algorithmes d'OCR.

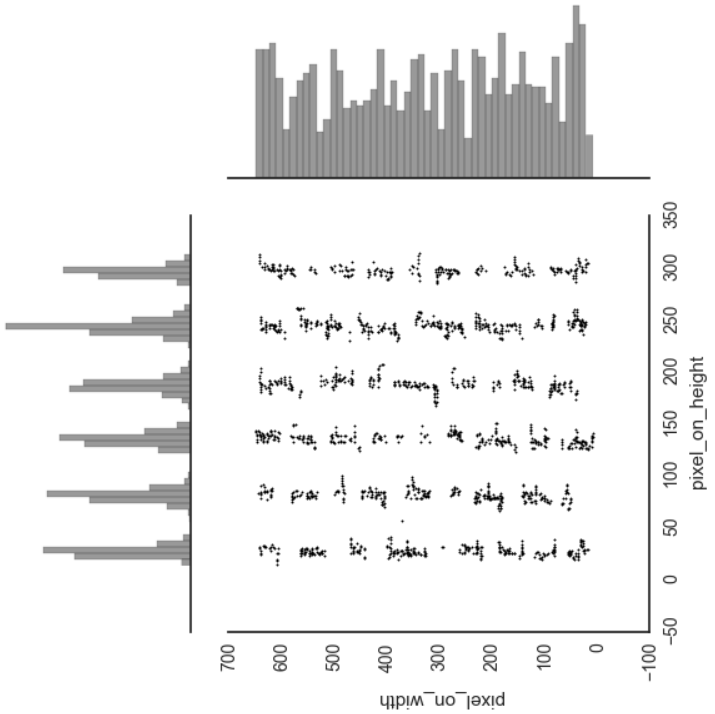


Figure 63.: Distribution des pixels dans une image sur fond blanc contenant de l'écriture manuscrite

Comme le détaille Dupré (2004) ou Smith (2007), les algorithmes commencent généralement par décomposer les zones de texte en lignes puis en mots. Une méthode efficace consiste en effet à étudier à partir de l'image contenant du texte la distribution des pixels sur les lignes puis de réitérer l'étude, une fois les lignes détectées, sur les colonnes afin d'en extraire les mots comme l'illustre la Figure 63. De manière concrète, une estimation par histogramme des distributions puis l'établissement de seuils à partir des quantiles empiriques Dupré (2004) permet d'isoler les lignes et les mots de manière relativement simple. On obtient ainsi la segmentation illustrée en image 65 et 64

Figure 64.: Segmentation des mots dans une ligne de texte préalablement extraite

C'est alors sur ces échantillons de mots isolés que nous avons calibré notre algorithme d'extraction F en utilisant des réseaux de neurones profonds. Ce choix de modèle a notamment été motivé par les performances [Szegedy et al. 2016] relevées dans la littérature [Graves and Schmidhuber 2010]. Nous souhaitions ainsi mieux comprendre ces modèles.

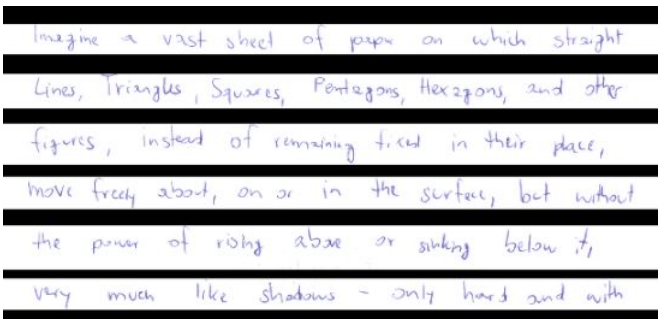


Figure 65.: Segmentation des lignes de textes

4.2.2 *Architecture du réseau de neurones*

Comme indiqué au chapitre 2, calibrer un réseau de neurones se ramène à optimiser les différents poids qui le constituent en minimisant une fonction de coût empirique (cf le chapitre 3). Cependant, dans notre cas d'application, aucun théorème en dehors du théorème de convergence universelle de Cybenko [Cybenko 1989] ne nous renseigne quant à la meilleure architecture à adopter. Notre approche a donc été progressive et bien qu'heuristique, inspirée de la littérature.

Au fondement de l'OCR appliquée à l'écriture manuscrite, réside la volonté de répliquer l'action du cerveau humain lors de la lecture. Dans l'apprentissage de cette dernière, nous commençons généralement par apprendre l'alphabet. Ainsi, une première idée consisterait à tenter de reconnaître dans un premier temps les lettres qui composent le mot. Cette méthode peut sembler prometteuse si l'on s'appuie sur les résultats[7] obtenus dans la détection des chiffres [Hinton and Salakhutdinov 2006]. Cependant, comme le mentionne Dupré (2004), dans l'écriture curviligne les lettres sont généralement attachées les unes aux autres et les dissocier avec une grande précision à partir des distributions de pixels est compliqué. Plutôt de que s'intéresser aux lettres, l'idée est alors de découper l'image contenant le mot en graphèmes. Autrement dit en régions ne respectant pas nécessairement la séparation des lettres. Les algorithmes présents dans la littérature –comme ceux proposés par Knerr et al. (2001), J. C. Simon (1992) ou Lecolinet (1996)– sont surtout présents sous forme d'heuristiques. Ils ont la particularité de vouloir retranscrire une extraction séquentielle de l'image comme notre œil pourrait le faire lors d'une lecture. La transposition de cette transformation dans le cadre des réseaux de neurones nous a donc amené à considérer des couches de convolution dans notre modèle.

En outre, la convolution [LeCun et al. 1998] nous permet d'apporter une nouvelle représentation de l'image et ainsi passer de l'espace initial I_k à un espace de plus grande dimension. Pour faire le lien à ce qui a pu être introduit dans les chapitres précédents sur les réseaux de neurones, la convolution est généralement en *deep learning* une transformation ϕ_K associée à une matrice que l'on notera K (appelé également filtre ou noyau de convolution) de taille $k_\phi \times k_\phi$ avec $k_\phi \in \mathbb{N}^*$. Cette transformation s'applique à chacune des composantes d'une image I_k autrement dit si X est une matrice représentant une image noir et blanc (i.e. $k = 1$) de taille $n \times m$, la convolution est l'application

$$\phi_K : \mathbb{R}_{n\times m} \to \mathbb{R}_{n'\times m'}$$

7 Les précisions des réseaux de neurones atteignent en effet les 99,5% de précision.

qui va associer à la matrice X, une matrice X' de taille $n' \times m'$ dépendant du paramètre p appelé *stride*. Le *stride* p est le pas de fenêtre glissante sur lequel le filtre K est appliqué. Ainsi, on a ainsi

$$n' = \frac{n-k}{p} + 1$$

et

$$m' = \frac{m-k}{p} + 1.$$

Finalement la matrice X' est définie par $\forall (i,j) \in [\![1, n']\!] \times [\![1, m']\!]$

$$X'_{i,j} = \sum_{l=i}^{i+k} \sum_{m=j}^{j+k} K_{l,m} X_{l,m} + c.$$

De fait, en terme d'apprentissage, les poids $K_{1,1}, K_{1,2}, ..., K_{k,k}, c$ sont les paramètres à optimiser dans notre réseau de neurones.

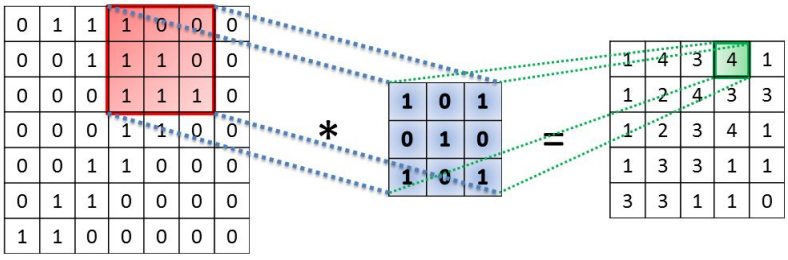


Figure 66.: Illustration d'une opération de convolution

Différentes couches de convolution ont ainsi pu être testées pendant nos essais (les meilleurs résultats obtenus sont présentés plus loin dans le tableau 4.2.4.2). Néanmoins, la considération des graphèmes à eux seuls n'est pas suffisante. En effet, lors de la lecture d'un texte, l'ordre des lettres conditionne le sens du mot. La dépendance temporelle est donc importante. En définissant un mot m comme une séquence finie de vecteurs de dimension fixe $D \in \mathbb{N}^*$, soit $m \in (\mathbb{R}^D)^{\mathbb{N}}$, une modélisation temporelle est alors possible. Dupré (2004), par exemple s'intéresse à l'utilisation de Chaines de Markov Cachés (HMM : Hidden Markov Model). Cette notion de dépendance temporelle est transcrite au travers des réseaux de neurones récurrents [Pineda 1987] détaillé dans l'annexe A.3 page 299. En outre, une structure de réseau de neurones récurrent bidirectionnels permettrait de tenir compte du graphème précédent et suivant afin d'inférer la lettre potentiellement qu'il contient. Cependant, afin d'entraîner un tel réseau à partir d'une base de données contenant des images de mots (comme celle introduite précédemment), il faudrait être en mesure de non pas associer un mot à chaque image mais la valeur de la ou les lettres que le graphème représente. Cette partie constitue une problématique en termes de labellisation qui tout aussi (si ce n'est plus) coûteuse en temps que de labelliser

manuellement tous les documents numérisés que nous disposons. Cela rendrait alors caduque notre approche.

Dans ses publications [Graves et al. 2012], [Graves et al. 2006], [Graves et al. 2009]), Alex Graves introduit une nouvelle architecture de réseau de neurones et plus particulièrement une nouvelle couche de sortie (*output layer*) appelée *Connectionnist Temporal Classifcation*, (CTC). Cette dernière permet de pouvoir considérer l'image d'un mot et le texte qu'il contient en tant que base d'apprentissage. Plus précisément cette couche peut être assimilée à une transformation inverse permettant de repasser d'une séquence graphèmes à un mot m appartenant à un dictionnaire $\mathcal{D} \sqsubset \mathbb{R}^D$, ensemble fini de mots. En effet, plutôt que de chercher à segmenter l'image en lettres (ou graphèmes) puis à prédire le label -i.e. la lettre- de chacun de ces segments (via un réseau convolutionnel suivi d'un réseau récurrent par exemple), Graves et al. (2006) propose d'utiliser directement l'image d'un mot comme entrée à un réseau de neurones récurrent et une séquence de lettres comme sortie cette dernière est ensuite post-traitée de manière à se ramener à la variable y initiale à expliquer. Le schéma 68 page 192 illustre la sortie obtenue par une telle couche de sortie.

En définitive nous avons choisi d'adopter la structure suivante :

- En entrée, les éléments sont des images de même taille contenant toutes un mot.
- Les premières couches du réseau sont des couches de convolution permettant d'extraire depuis l'image initiale de nouvelles variables et ainsi étendre la notion de découpage en graphème.
- Afin de tenir compte de la potentielle dépendance temporelle entre les différentes zones de l'image initiale, les sorties de la couche convolutionnelle vont alimenter différentes couches récurrentes bidirectionnelles.
- Enfin, une couche de sortie CTC permet de revenir à l'espace des mots et de calculer notre métrique de précision.

Le choix du nombre de couches dans la partie convolutionnelle et récurrente provient de différents tests et s'inspire également de différentes structures qui ont pu être rencontrées dans d'autres expérimentations de *deep learning*. Le choix de la structure finale est présentée dans le tableau 4.2.2 :

Composition de la couche	**Dimension**
Image contenant un mot	128×64
Convolution à noyaux de taille 3×3	$128 \times 64 \times 16$
Max pooling de taille 2×2	$64 \times 32 \times 16$
Convolution à noyaux de taille 3×3	$64 \times 32 \times 16$
Max pooling de taille 2×2	$32 \times 16 \times 16$
Redimensionnement	32×256
Couche *dense*	32×32
GRU bidirectionnel	$2 \times 32 \times 512$
Somme	32×512
GRU bidirectionnel	$2 \times 32 \times 512$
Concatenation	32×1024
Couche *dense*	32×64
Couche CTC	$\lvert\mathcal{D}\rvert$

Table 20.: Structure finale du réseau de neurones retenu dans notre cas d'application

4.2.3 *Zoom sur la dernière couche de sortie CTC (Connectionist Temporal Classification)*

Considérant l'ensemble A des caractères pouvant être reconnus (et donc prédits à la sortie du réseau récurrent), on définit l'ensemble $L = A \cup \{\emptyset\}$ où $\emptyset$ indique un champ vide. La couche CTC peut ainsi se définir comme la succession d'une couche *dense* (i.e. une couche de neurones linéaires comme vu au chapitre 2 et une fonction d'activation *Sigmoïde*. La dimension de cette couche de sortie est égale à $\mid L \mid$ soit le nombre de classes (ou de caractères) pouvant être reconnus. Cette couche, positionnée en sortie des résultats des couches de neurones récurrents permet ainsi d'affecter à chaque composante $t \in [1, T]$ de la séquence temporelle fournie par la couche récurrente une probabilité d'appartenir à la classe $l \in L$ que l'on note y_l^t. On obtient une matrice de probabilité de taille $T \times L$ comme illustrée sur la Figure 67.

En supposant l'indépendance des probabilités y_l^t conditionnellement à l'entrée X[8], on obtient alors la distribution suivante pour la séquence $\pi \in L^T$:

$$p(\pi|x) = \Pi_{t=1}^{T} y_{\pi_t}^t$$

avec π étant la séquence constituée en choisissant la classe l à l'instant t. De manière simple, on peut choisir à chaque instant t la classe l ayant la meilleur probabilité.
Considérons m un mot, connu dans la phase d'entraînement, cette séquence de caractères est alors à relier à la séquence de sortie du réseau récurrent. En effet, la séquence

8 On rappelle que par simplification, on considère que notre image peut être assimilée à une unique matrice que l'on note X. Cette dernière peut être vue comme une composante de l'image I_k.

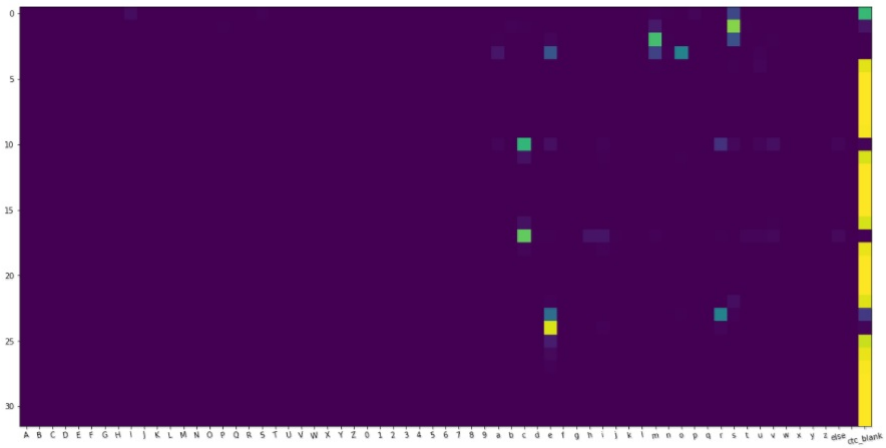


Figure 67.: Matrice de probabilités obtenue en sortie de la couche CTC avec comme entrée une image de l'écriture manuscrite du mot 'marche'. En abscisse sont indiquées les classes et en ordonnée le temps t, la couleur représente la valeur de la probabilité, la couleur violette correspondant à la valeur 0.

de sortie du réseau récurrent est à différencier d'une séquence de lettres (soit du mot) puisqu'il contient encore des éléments non contenus dans le mot ($\emptyset$ par exemple, ou des doublons de lettres) et qu'il est de manière générale, de taille supérieure à la séquence de lettres à prédire.

C'est pourquoi on définit une fonction $\mathbf{M}$ qui permet de relier une séquence m de lettre à une séquence π quelconque. Cette fonction relie l'ensemble des sorties du réseau récurrent possibles à l'ensemble des mots $\mathcal{D}$.

On obtient, par exemple, $\mathbf{M}(a\_\_\_ab\_) = \mathbf{M}(\_aa\_\_\_aab\_\_\_) = aab$. Puisque tous les chemins sont mutuellement exclusifs, c'est-à-dire qu'une séquence π est transformée en au maximum une seule séquence de lettres, on peut définir la probabilité

$$p(l|x) = \sum_{\pi \in \mathbf{F}^{-1}(l)} p(\pi|x)$$

qui est la probabilité de reconnaître la séquence de lettre m connaissant X.
On peut alors entraîner notre modèle avec comme objectif la maximisation de la log-vraisemblance du modèle c'est-à-dire la maximisation de la log-probabilité de $p(l|x)$. En pratique le calcul de cette probabilité est coûteux du fait que la somme sur toutes les séquences peut être grande. Un algorithme de type *forward-backward* similaire à celui utilisé dans les méthodes de reconnaissance par HMM [Dupré 2004] permet d'accélérer l'optimisation, en utilisant notamment la descente de gradient.

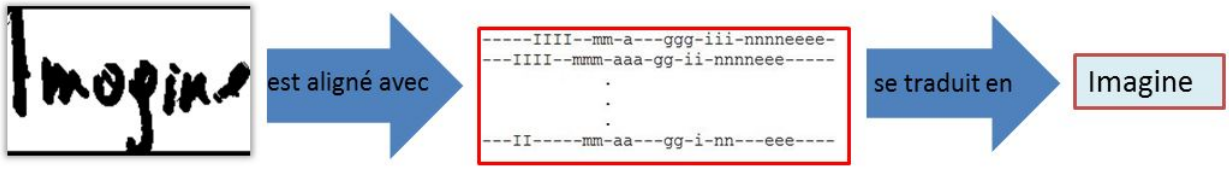


Figure 68.: Schéma de la mise en correspondance entre l'image d'entrée, la sortie de la couche récurrente et la prédiction

4.2.4 *Présentation des résultats et perspectives*

L'architecture définie précédemment comporte ainsi près de 5 millions de paramètres réels à calibrer. L'annexe page 303 apporte plus de précisions quant aux choix techniques et technologiques qui ont permis de calibrer notre modèle. Par ailleurs, les bases de données que nous avons utilisé proposent certains *benchmarks* obtenues par d'autres modèles. Nous avons ainsi souhaité nous y référer afin de situer nos performances.

4.2.4.1 *Mesures d'évaluation*

Si nous avons précédemment défini une mesure de précision associée à une fonction d'OCR F, dans le cas de l'apprentissage de notre modèle, nous avons utilisé des métriques légèrement différentes permettant de prendre en compte la bonne reconnaissance ou non des caractères qui composent les mots extraits.

Afin de mesurer la qualité des modèles, on définit alors deux mesures :

Définition 4.2. Soit C_1 et C_2, deux séquences de caractères contenant chacune un unique mot d'un dictionnaire $\mathcal{D}$. On considère ainsi que $|C_1| = |C_2|$ et que potentiellement un caractère peut être vide de sorte que les mots que C_1 et C_2 contiennent ne sont pas nécessairement de la même longueur. On définit alors la distance d'édition Δ comme :

$$\Delta(C_1, C_2) = \sum_{(c_1,c_2)\in(C_1,C_2)} \mathbb{1}\{c_1 \neq c_2\}.$$

Autrement dit $\Delta(C_1, C_2)$ représente le nombre de tranformations nécessaires afin de rendre C_1 identique à C_2

Définition 4.3. Soit $k \in N^*$, soit $n \in \mathbb{N}^*$, soit $(I_{k,i}, C_i)_{i=1,..,n} \in (\mathcal{I}_k \times \mathcal{C})^n$, n couples d'images contenant une chaine de caractère C_i représentant un mot. On note pour tout $i^=1, .., n$, $|C_i|$ le nombre de caractères que représente l'image $I_{k,i}$. Soit F, une fonction

d'OCR. On définit alors la précision sur les caractères, appelée *Character Error Rate (CER)* :

$$CER = \frac{1}{n}\sum_{i=1}^{n}\frac{\Delta(C_i, F(I_{k,i})}{|C_i|}$$

et la précision sur les mots, appelée *Word Error Rate (WER)*:

$$WER = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{C_i \neq F(I_{k,i})\}.$$

4.2.4.2 *Résultats*

Les travaux de Graves et al. (2012), démontrent que les réseaux de neurones amènent à de meilleures performances que l'utilisation des modèles HMM. Les bases de données utilisées et introduites dans la section précédente (CVL [Kleber et al. 2013] et RIMES [Grosicki and El Abed 2009]) proposent par ailleurs des échantillons de test permettant ainsi de comparer les résultats de nos modèles. Cet échantillon représente environ 1000 images qui n'ont donc pas été utilisées lors de l'apprentissage. Le tableau 21 présentent les différents résultats obtenus par notre modèle sur l'échantillon de test.

Modèles	**CER (%)**	**WER (%)**
LSTM 30 [Pham et al. 2014]	14.72	42.03
LSTM 50 [Pham et al. 2014]	15.11	42.62
LSTM 100 [Pham et al. 2014]	15.79	44.37
LSTM 200 [Pham et al. 2014]	**14.68**	42.07
Our neural network	19.0	44.5
Our neural network with correction	17.11	**36.3**

Table 21.: Résultats obtenus sur la base de test

Les résultats obtenus sur la base de test Rimes sont assez satisfaisants (19 en CER et 44.5 en WER), bien que plus faibles que ceux obtenus par Pham et al. (2014). En effet, bien que le taux d'erreur WER semble élevé, il est néanmoins à noter que l'on compte comme incorrecte une prédiction dès lors qu'au moins un des caractères n'a pas été prédit correctement. Le CER moyen de 19 indique alors qu'en moyenne 81% du mot prédit est néanmoins correct. Par ailleurs si on regarde un échantillon de résultats comme illustré en Figure 69, on remarque que les erreurs commises concernent des images où une lecture humaine est parfois difficile.

Figure 69.: Résultats de la reconnaissance de l'écriture manuscrite par les réseaux de neurones

L'utilisation d'un correcteur orthographique, permet alors de corriger certaines fautes ce qui permet d'obtenir score WER meilleur que celui constaté dans la littérature. Ce correcteur s'appuie sur un dictionnaire et effectue des corrections par similarités (au sens de la distance de Levenhstein).

On rappelle que notre objectif initial était d'être en mesure d'analyser automatiquement des documents contenant de l'écriture manuscrite. Parmi ces documents, les constats amiables automobiles ont particulièrement retenu notre intérêt. Il représente en effet un document fréquemment reçu par les assureurs et qui nécessite en général une saisie manuelle du document par un agent. Automatiser leur analyse aurait en effet pu permettre un fluidification du processus d'indemnisation des sinistres. Nous avons donc tenté d'appliquer notre algorithme sur quelques échantillons de constats. la Figure 70 présente un exemple.

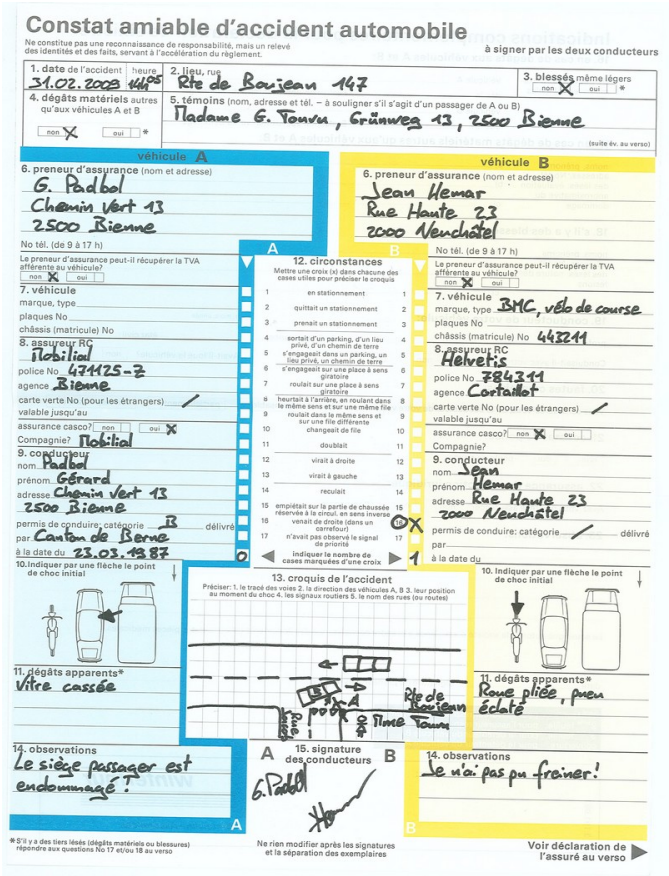


Figure 70.: Constat amiable rempli

Nous avons appliqué à cette image certaines heuristiques afin d'isoler dans un premier temps l'écriture manuscrite du formulaire initial. Le résultat est présenté en Figure 71. Plus particulièrement, constatant que l'écriture manuscrite est en général plus épaisse que l'écriture tapuscrite. Une idée consiste alors à appliquer un filtre par exemple de *max pooling* sur une image convertie préalablement en noir et blanc. Il en ressort alors les éléments qui nous intéressent.

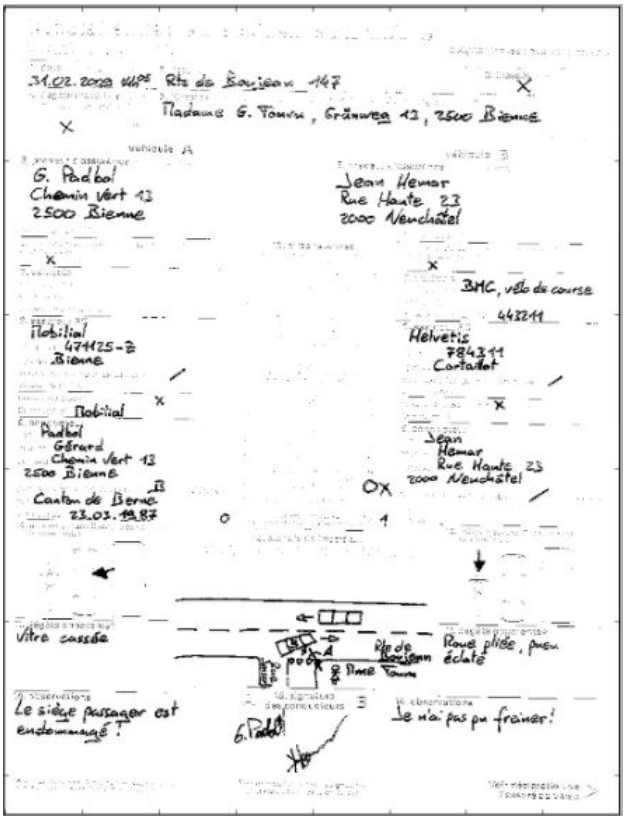


Figure 71.: Exemple de constat retraité pour un seuil donné

Après découpage des zones et la mise au bon format, nous avons ensuite appliqué notre algorithme de reconnaissance de mots préalablement calibré. Les résultats sont assez discutables. La Figure 72 illustre en effet les résultats obtenus sur notre échantillon. De nombreux mots sont alors peu reconnus voir totalement éloignés de ce que l'on pourrait lire. Ce résultat illustre un problème traité par les recherches sur les réseaux de neurones profonds qui est celui du transfert de connaissance. En effet, notre réseau a été entraîné sur une base de données contenant un vocabulaire littéraire et peu commun avec celui que l'on pourrait trouver dans des constats amiables d'assurance automobile. Ce résultat illustre ainsi la faible capacité de notre réseau à reconnaître des termes qu'il n'a pas vu par le passé. Or, on pourrait penser que la reconnaissance d'écriture est censée être indépendante des termes que l'on utilise pour "apprendre" l'extraction. Cette limitation est ici structurelle. En entraînant notre modèle à reconnaître un mot, nous nous sommes focalisés exclusivement sur cette tâche.

Figure 72.: Résultats obtenus sur les mots extraits du constat

Une solution d'amélioration serait alors d'enrichir notre base de données en s'assurant d'avoir un vocabulaire d'images plus riche et contenant une grande diversité d'écritures. C'est ainsi que nous avons considéré l'utilisation de bases de données génératives [Graves 2013]. Néanmoins, lors de ces recherches, la digitalisation des constats amiables fut annoncé pour les années suivantes, rendant moins intéressants nos travaux initialement motivés par le développement d'une nouvelle offre de service. Il a donc été décidé d'orienter nos recherches sur le traitement automatique du texte tapuscrit. Or, les méthodes étudiées en introduction comme *Tesseract* sont aujourd'hui suffisamment performantes pour répondre aux besoins industriels. C'est la raison pour laquelle nous nous sommes donc finalement intéressés aux méthodes d'analyses sémantiques appelées également *text mining* en anglais.

4.3 LES TECHNIQUES D'ANALYSE SÉMANTIQUE : LE *text-mining*

Les techniques d'extraction de texte à partir d'images à elles seules n'ont que peu d'intérêt (en dehors d'être en mesure d'indexer plus facilement les documents) si nous ne les associons pas à des techniques d'analyse automatique de texte. Ces techniques connaissent depuis ces dix dernières années un très fort intérêt notamment au travers des réseaux de neurones profonds. La section qui suit expose ainsi différentes techniques les plus couramment utilisées pour l'analyse sémantique. Plus particulièrement, une librairie ouverte aux collaborateurs de Milliman a pu être développée dans le cadre de ces travaux d'exploration. Le but était de faciliter l'accès aux techniques et d'établir un cadre harmonisé entre les différentes librairies pré-existantes.

4.3.1 *Le Natural Language Processing (NLP)*

Les premières extractions automatiques d'information à partir de texte remontent aux années 1970 avec les travaux de Dejong (1979). Désignées le plus souvent sous le terme anglais de *Natural Language Processing* (NLP), les analyses sémantiques se sont par la suite développées dans les années 1990 [Grishman and Sundheim 1996] avec la création de la *Message Understanding Conference* (MUC). Si les recherches se sont intensifiées depuis, la MUC propose initialement d'extraire des informations linguistiques en utilisant des templates dans lesquelles des champs peuvent être renseignés. D'autres sont alors automatiquement remplis en fonction des informations extraites depuis les valeurs indiquées par l'utilisateur. Dans cette approche, la structure de la réponse est supposée connue. Ainsi dans certains champs, un simple nom commun est attendu alors que dans d'autres, une phrase ou des adjectifs peuvent être renseignés. L'extraction d'informations à partir de templates pré-suppose donc que l'on connaisse par avance la structure de l'information qu'il va falloir extraire (par exemple un code postal dans un adresse, un nom d'entité ou encore une relation). Cette approche est ainsi très limitée dans de nombreuses situations. Des alternatives moins contraignantes sont apparues afin d'être en mesure d'extraire de l'information sur du texte libre. Particulièrement favorisé par le développement d'internet et la croissance du nombre de sites web regorgeant d'informations textuelles [Aggarwal and Zhai 2012], les chercheurs s'intéressent dans un premier temps à exploiter le texte semi-structuré sous le format HTML [Banko et al. 2007]. On distingue alors deux tâches majeures qui s'inscrivent au fondement du NLP [Jiang 2012] :

- La reconnaissance de nom d'entité désignée la plupart du temps par l'acronyme anglais NER (*Name Entity Recognition*) : ces techniques s'intéressent à extraire, à partir d'un texte, la valeur d'un nom commun ou propre mais s'étend également à la reconnaissance de valeurs de métriques (date, prix, etc.).

- L'extraction de relations, faisant référence aux techniques tentant de déterminer les liens logiques entre les différents mots d'un texte. A partir d'un texte brut ou semi-structuré, l'enjeu consiste à retrouver les relations grammaticales entre les termes.

Les paragraphes qui suivent introduisent alors de manière synthétique, les différents algorithmes majoritairement utilisés en NLP. Les définitions suivantes sont alors utilisées afin d'expliquer les modèles.

Définition 4.4. On appellera "mot" noté m une séquence finie de vecteurs de dimension $D \in \mathbb{N}^*$ Ainsi,

$$m \in (\mathbb{R}^D)^{\mathbb{N}}.$$

Chaque composante sera appelée "caractère".

Définition 4.5. On appellera "dictionnaire", le sous-ensemble $\mathcal{D} \sqsubset (\mathbb{R}^D)^{\mathbb{N}}$.

Définition 4.6. On définit un "document" d de longueur n_d comme une suite finie de mots m soit

$$d \in \mathcal{D}^{\mathbb{N}} \; et \; d = (m_i)_{i \in [[1,n_d]]}$$

Définition 4.7. On appelle corpus, noté $\mathcal{C}_n$ une séquence finie de $n \in \mathbb{N}^*$ documents. Ainsi,

$$\mathcal{C}_n = (d_i)_{i \in [[1,n]]}$$

4.3.1.1 *La reconnaissance de nom d'entité*

La manière la plus simple d'implémenter la reconnaissance de noms d'entité est sans doute l'utilisation de règles. Lorsqu'un motif est détecté, une règle est ainsi appliquée. L'utilisation des expressions régulières [Califf and Mooney 1999] permet ainsi d'extraire un information textuelle qui suit un format particulier. Par exemple, "Mr/. (.*)" permet de s'intéresser à tous les textes commençant par "Mr" et de s'intéresser au groupe qui le suit qui est censé représenter le nom de la personne. Des règles plus précises permettent de spécifier les valeurs de caractères autorisés avec lesquelles le motif est censé correspondre. Les règles par expressions régulières sont aujourd'hui couramment utilisées en industrie et permettent de répondre à de nombreux sujets.

D'autres approches existent comme par exemple des approches d'apprentissage statistique. Dans ce cadre, l'idée générale est d'être en mesure de pouvoir affecter à tout mot m composant un document d, une catégorie y indiquant si le mot représente une entité particulière à extraire ou non. Plus exactement, considérons un corpus de $n \in \mathbb{N}^*$ documents $C_n = (d_i)_{i \in [[1,n]]}$ que l'on supposera tous de longueur constante égale à $n_d \in \mathbb{N}^*$ mots appartenant à un dictionnaire $\mathcal{D}$. Ainsi, on a

$$\forall i \in [[1, n]], \; d_i = (m_l)_{l \in [[1,n_d]]} \in \mathcal{D}^{n_d}$$

Dans le cadre de la reconnaissance d'entité par apprentissage statistique, on suppose également que l'on dispose pour chaque document d_i des caractéristiques associées aux mots qui les composent. On notera ainsi

$$\forall i[[1,n]],\ y_i = (y_i^1, .., y_i^{n_d}) \in \mathcal{E}^{n_d}$$

Où $\mathcal{E}$ désigne l'espace fini des entités par exemple si un mot est un nom, une date, etc. Autrement, dit, l'apprentissage statistique aborde la question de la reconnaissance d'entités sous la forme d'un problème de classification. Comme l'indique Jiang (2012), les labels associés aux différents mots suivent généralement une convention appelée BIO [Rau 1991].

Steve	Jobs	was	a	co-founder	of	Apple	Inc.
B-PER	I-PER	O	O	O	O	B-ORG	I-ORG

Figure 73.: Exemple d'un vecteur de labels y_i suivant la convention BIO associé à un document d_i

Ainsi, si l'on considère les variables aléatoires y et d, une solution proposée pour définir un algorithme de NER, est de modéliser la probabilité $\mathbb{P}(y|d)$ soit indirectement d'être en mesure de calculer la probabilité jointe $\mathbb{P}(y, d)$. Une façon de résoudre ce problème est de considérer un processus de Markov [Rabiner 1989] où les valeurs cachées que peuvent prendre les composantes de $y = (y^1, ..., y^{n_d})$ ne dépendent que des composantes ultérieures. Ainsi, Bikel et al. (1997) proposent par exemple de supposer que pour tout[9] $l \in [[1, n_d]]$:

- Chaque composante y^l n'est conditionnée que par la valeur de la composante précédente y^{l-1} et du mot associé qui le précède m^{l-1}.
- Dans le cas d'un nom d'entité se décomposant sur plusieurs mots, si m^l est le premier mot de l'entité, sa valeur est alors conditionné par y^l et y^{l-1}
- si m^l est à l'intérieur d'une séquence de mots définissant l'entité, sa valeur est alors conditionné par le mot précédent m^{l-1} uniquement.

Ainsi, étant donné l'observation du corpus de document $C_n = (d_i)_{i\in[[1,n]]}$ et l'ensemble des caractéristiques associées $(y_i)_{i\in[[1,n]]}$, Bikel et al. (1997) proposent une approche d'apprentissage supervisé afin notamment d'être en mesure de déterminer pour chaque composante $l \in [[1, n_d]]$, la probabilité $\mathbb{P}(y^l = e_1|y^{l-1} = e_2, m^{l-1} = w)$ que cette dernière soit associée à une classe conditionnellement à l'observation de la classe de

9 Les effets de bords au premier indice sont traité grâce à l'ajout de balises en début de chaque document.

la composante précédente et du mot associé. Cette dernière est par ailleurs calculée empiriquement par :

$$\frac{\sum_{i=1}^{n} \mathbb{1}\{y_i^l = e_1, y_i^{l-1} = e_2, m_i^{l-1} = w\}}{\sum_{i=1}^{n} \mathbb{1}\{(y_i^{l-1} = e_2, m_i^{l-1} = w\}}$$

où e_1 et e_2 sont deux classes de l'ensemble $\mathcal{E}$, w la valeur possible d'un mot. Cette approche a l'avantage, une fois calibrée, de proposer un modèle génératif de documents.

4.3.1.2 *L'étude des relations*

Une seconde catégorie de méthodes consiste à définir des relations entre les entités extraites. Par exemple, dans la phrase "*Antoine est le président du Data Science Game*", l'extraction d'entités amènerait à définir "*Antoine*" comme un nom propre et "*Data Science Game*" comme un nom d'entreprise par exemple. Cependant, l'extraction de nom d'entité ne permet pas d'inférer qu' "*Antoine*" est le dirigeant de "*Data Science Game*". Cette tâche d'étude de relation peut être abordée comme une problématique de classification entre deux entités co-occurentes. Ainsi, Jiang and Zhai (2007) proposent une approche systématique en représentant la relation entre les entités par un graphe dirigé. Ils introduisent le graphe orienté $G = (V, E, A, B)$ où V représente l'ensemble des nœuds, E les connexions et A et B un ensemble de fonctions attribuant certaines propriétés aux nœuds. Pour tout nœud $m \in V$, $A(m) = \{a_1, a_2, ..., a_{|A(m)|}\}$ est l'ensemble des attributs associé au nœud v où $a_i \in \Sigma$ et Σ, un ensemble contenant toutes les valeurs possibles (type d'entité, la forme du mot, etc.). L'application $B : V \to \{0, 1, 2, 3\}$, indique quant à elle, la relation entre l'attribut et le nœud $m \in V$. Cette représentation permet en outre de retranscrire une phrase ainsi que les relations entre les termes qui la constituent. Elle permet également d'en extraire des sous-structures et donc des informations pertinentes.

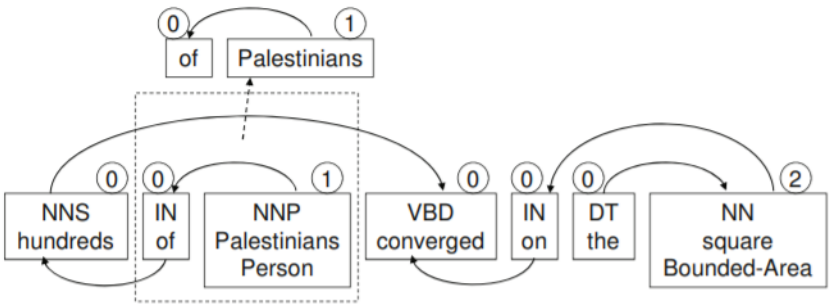


Figure 74.: Exemple d'un arbre de dépendances suivant le formalisme introduit par Jiang and Zhai (2007)

Toujours en conservant cette idée de représentation graphique des connexions entre les entités et de pouvoir extraire des sous-groupes d'entités reliées, des méthodes

à noyaux s'appuyant sur des arbres sont proposées par Zelenko et al. (2003). Ces méthodes seront notamment étendues avec l'application d'arbres convolutionnels [Zhang et al. 2006]. Comme décrit précédemment dans ce chapitre, les convolutions permettent une représentation d'un vecteur dans un nouvel espace. Dans un arbre convolutionnel, chaque dimension du vecteur peut alors être associée à un sous-ensemble de l'arbre et Jiang and Zhai (2007) expose alors des exemples d'utilisation de cette méthode.

Si les méthodes ont beaucoup évolué depuis le début des années 2000, la volonté d'établir des relations entre les mots -et ainsi adopter une approche grammaticale de l'analyse sémantique- reste d'actualité. Plus récemment, Speer et al. (2017) proposent une représentation générale de l'écriture et notamment de la langue en utilisant également la théorie de graphes. Le résultat, appelé *ConceptNet* [10] est ainsi un graphe logique qui connecte les mots ou des phrases. Sa construction s'appuie sur des données apportées par des experts et d'autres issues de questionnaires. La motivation est ainsi de fournir une nouvelle représentation de la langue et ainsi améliorer la compréhension du langage [Speer et al. 2017]. La constitution de ce graphe a été largement favorisée par le développement des techniques de parallélisation des calculs et la quantité de données disponibles. Ainsi les nœuds (ou les mots) sont mis en relation par des connections comme : "*Est utilisé pour*", "*Est capable de* ", "*Est un*". La Figure 75 en illustre un extrait.

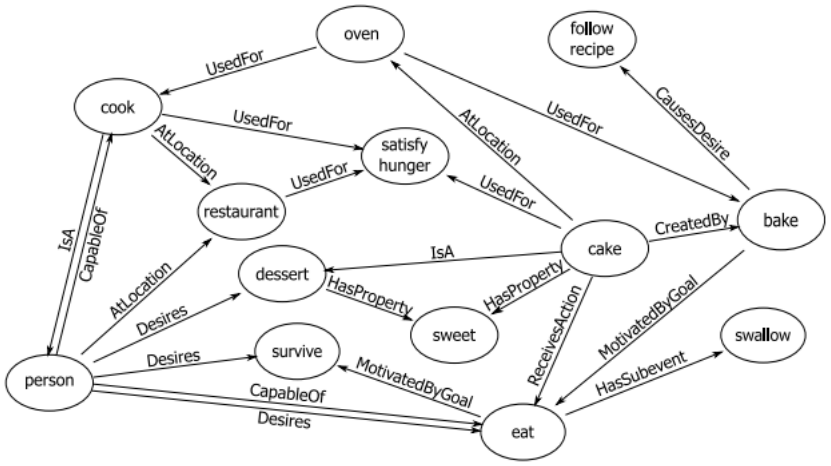


Figure 75.: ConceptNet

Ainsi le graphe permet d'assurer un lien logique entre chaque mot. C'est en s'appuyant sur cette représentation qu'a été conçu *ConceptNet-Numberbatch* [Speer et al. 2017] que nous détaillons dans les sections ci-après.

10 `http://conceptnet.io/`

4.3.2 *L'approche statistique*

Si l'extraction d'information à partir de texte libre permet de répondre à la détection de noms d'entité ou l'étude des relations entre les termes, l'utilisation du texte peut également se faire afin de résoudre des problèmes de régression ou de classification plus standards [Aggarwal and Zhai 2012]. Plutôt que de considérer les mots comme des entités mises en relation par des attributs, une possibilité est alors de simplement considérer les mots comme des occurrences d'une variable aléatoire. De manière similaire à la méthode introduite dans l'approche statistique permettant la reconnaissance d'entités, les documents sont alors décrits par des statistiques [Nenkova and McKeown 2012] sans tenir compte néanmoins de leur ordre. L'idée est ainsi de considérer que des mots apparaissant fréquemment ensemble dans un document révèlent une information.

Ainsi une première représentation statistique du texte consiste à représenter chaque document d dans la base canonique d'un dictionnaire $\mathcal{D}$. Chaque dimension représente alors un mot du dictionnaire. Dans cette représentation, appelée "sacs de mots", l'idée consiste à simplement sommer les vecteurs canoniques associés aux mots composant la phrase. Dans la pratique, le dictionnaire $\mathcal{D}$ est rarement défini comme l'intégralité des mots d'une langue. Il est constitué des mots observés dans le corpus C_n de documents utilisés dans l'étude. Ainsi, si l'on possède qu'un seul document qui contient la phrase "*Je m'appelle Antoine*", le dictionnaire associé sera $\{je, m, appelle, Antoine\}$. Selon ce même exemple, on représenterait chaque mot dans un espace de dimension 4 et tout mot non présent dans le dictionnaire serait affecté au vecteur nul. La phrase "*Je m'appelle Adrien*" serait alors représentée par le vecteur $[1, 1, 1, 0]$

Cette représentation en "sac de mots" ou Bag of Words (BoW) en anglais, si elle est simple, présente l'inconvénient de ne pas être parcimonieuse. En effet, dans une phrase, nous employons de manière générale que peu de mots parmi ceux existant dans le dictionnaire. D'autre part, les articles, mots de liaison ou termes courants (appelés *stopwords*) sont souvent sur-représentés par cette méthode. Pour faire face à ce dernier inconvénient, une heuristique consiste à fixer une certaines listes des mots indésirables et à les soustraire du dictionnaire.

Dans un corpus de documents spécifique, la fréquence de certains mots peut également être plus élevée que dans d'autres. Ainsi, une alternative à la représentation en sac de mots est celle qui consiste à ne plus simplement compter l'occurrence du mot dans le document mais également de tenir compte de son apparition dans le corpus.

Définition 4.8. On appelle fréquence d'un mot m dans un document d (*term frequency* ou tf) , la quantité $tf(m, d)$ définie comme :

$$tf(m, d) = \text{Le nombre de fois où le terme m apparaît dans le document d}$$

Définition 4.9. Soit $n \in \mathbb{N}^*$, soit C_n un corpus de document, on définit la quantité idf_m (*inversed document frenquency* ou idf) associée à un mot m la quantité :

$$idf_m = \log_{10} \frac{n}{df_m}$$

où df_m représente le nombre de documents où le mot m apparaît.

Un document peu alors être décrit dans la base canonique d'un dictionnaire non plus en sommant simplement les vecteurs représentant chaque mot mais en effectuant une somme pondérée à l'aide des poids appelés *tf_idf* [Gupta et al. 2007]. Sur chacune des $|\mathcal{D}|$ coordonnées du vecteur représentant un document $d = (m_1, ..., m_{n_d})$ on a donc un coefficient w_i^d donné par

$$\forall i \in |\mathcal{D}|,\ w_i^d = \begin{cases} 0 & \text{si le mot i du dictionnaire n'est pas dans d} \\ (1 + log(tf_{m,d})) \log_{10} \frac{n}{df_m} & \text{sinon} \end{cases}$$

La représentation en sac de mots et ses dérivées pondérées ont l'avantage de fournir une représentation vectorielle d'un document où chaque composante s'interprète comme un mot. Cette représentation permet non seulement d'utiliser différentes distances entre les documents mais également de traiter de nombreux sujets sous l'angle de l'apprentissage statistique supervisé. Une méthode particulièrement efficace dans de nombreuses situations est l'approche Bayesienne [Jurafsky 2011].

Ces algorithmes nommés *Naïve Bayes* sont supervisés et s'appuient sur le théorème de Bayes en faisant l'hypothèse "naïve" que les variables explicatives sont indépendantes. Soit une variable catégorielle y et les variables explicatives $x_1, ..., x_n$ (par exemple la représentation en sac de mots d'un document), le théorème de Bayes spécifie le calcul suivant :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

En utilisant, l'hypothèse d'indépendance i.e. $P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$, pour tout i cette relation est simplifiée à

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Puisque $P(x_1, \ldots, x_n)$ est constant, on peut donc utiliser la règle de classification suivante

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Ainsi, on peut définir $\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$ comme le label prédit en connaissant $x_1, .., x_n$, et on peut utiliser l'estimation du Maximum A Posteriori (MAP) afin d'estimer $P(y)$ et $P(x_i \mid y)$; le premier terme étant simplement la fréquence relative de la présence du label i dans les données.

Les classifieurs *Naïve Bayes* différent principalement par l'hypothèse faite sur la distribution $P(x_i \mid y)$. Par exemple, si on suppose cette distribution gaussienne, cela donne :

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Les paramètres σ_y et μ_y sont estimés par maximum de vraisemblance.

Les représentations en sac de mots permettent ainsi une utilisation de tout un panel d'algorithmes d'apprentissage. Cependant, cette représentation fréquentiste est peu parcimonieuse et ne tient pas compte des dépendances potentielles entre les différents termes d'un document.

4.3.3 *Le deep learning comme nouvelle référence*

Les représentations en sac de mots présentent l'inconvénient de manipuler des vecteurs en grande dimension. Mikolov et al. (2013) proposent alors d'apprendre de l'observation des corpus une représentation vectorielle qui permet de tenir compte du contexte de chaque mot mais également de réduire la dimension du vecteur. Cette méthode, se base ainsi sur l'utilisation de réseaux de neurones. Mikolov [Mikolov et al. 2013] introduit notamment deux structures : l'une permettant de prédire un mot connaissant son contexte (CBOW) et l'autre de prédire réciproquement le contexte conditionnellement au mot observé (Skip-gram).

4.3.3.1 *Zoom sur le modèle CBOW*

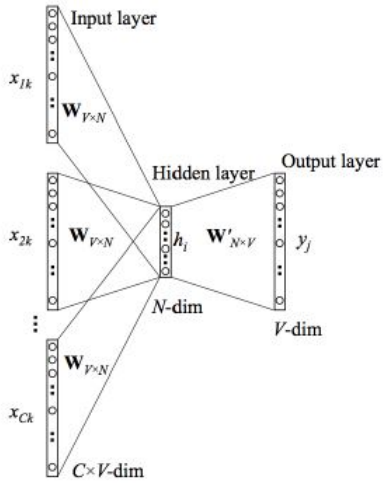


Figure 76.: CBOW, Efficient Estimation of Word Representations in Vector Space-Mikolov et al.

Dans l'apprentissage de ce modèle, une fenêtre de contexte $C \in \mathbb{N}^*$ est préalablement choisie. Ensuite, chaque document est décomposé en sous-documents de C mots. La résultante de cet échantillonnage est alors un nouveau corpus C_n de n documents de taille C représentés par un sac de mots. On notera par la suite V la dimension du dictionnaire (soit la dimension de chaque vecteur représentant un mot noté X sur le schéma 76). L'objectif du réseau est alors d'être en mesure de prédire un mot manquant connaissant son contexte (c'est-à-dire une fenêtre de mots telle que décrit précédemment).

Plus particulièrement, ce réseau débute par une projection. Elle consiste en une réduction de dimension via une matrice de taille $V \times N$ où N est la dimension de la projection que l'on choisit. Chaque vecteur d'entrée représentant les mots dans leur contexte est ainsi projeté selon cette matrice de passage. Les projections sont ensuite sommés en sortie de couche. Finalement, la sortie de la couche est suivie d'une couche *softmax* à V neurones permettant d'indiquer la probabilité qu'un mot du dictionnaire soit associé au contexte (noté sur le schéma y_i).

Après entraînement cette couche sert alors de matrice de passage entre la représentation en sac de mots et une représentation vectorielle de dimension plus petite. Cette transformation est souvent appelée *Word-to-Vec*.

4.3.3.2 *Zoom sur le modèle Skip-gram*

De manière réciproque, le modèle *Skip-gram* tente de prédire un contexte connaissant un mot.

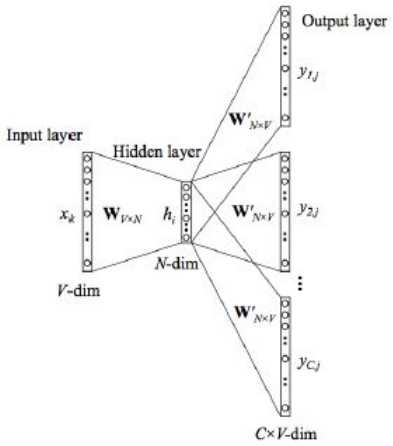


Figure 77.: Skip-gram, Efficient Estimation of Word Representations in Vector Space-Mikolov et al.

Ainsi l'entrée du modèle fournit la représentation canonique du mot X_t alors que la sortie tente de reconstituer le contexte à partir des poids de la couche de sortie $w_{t+1}, ..., w_{t+C}$. Ce type de réseaux permet par exemple de pouvoir inférer la suite d'un texte par exemple. De nos jours, de nombreux modèles entraînés sur des données volumineuses comme l'encyclopédie Wikipédia ou encore le réseau social Twitter sont mis à disposition de la communauté (Word2vec [Goldberg and Levy 2014], Glove [Pennington et al. 2014], etc.). Cette représentation vectorielle a pour avantage d'être plus parcimonieuse en plus de pouvoir tenir compte du contexte entre les différents mots c'est-à-dire de la proximité géométrique potentielle entre les mots co-occurants.

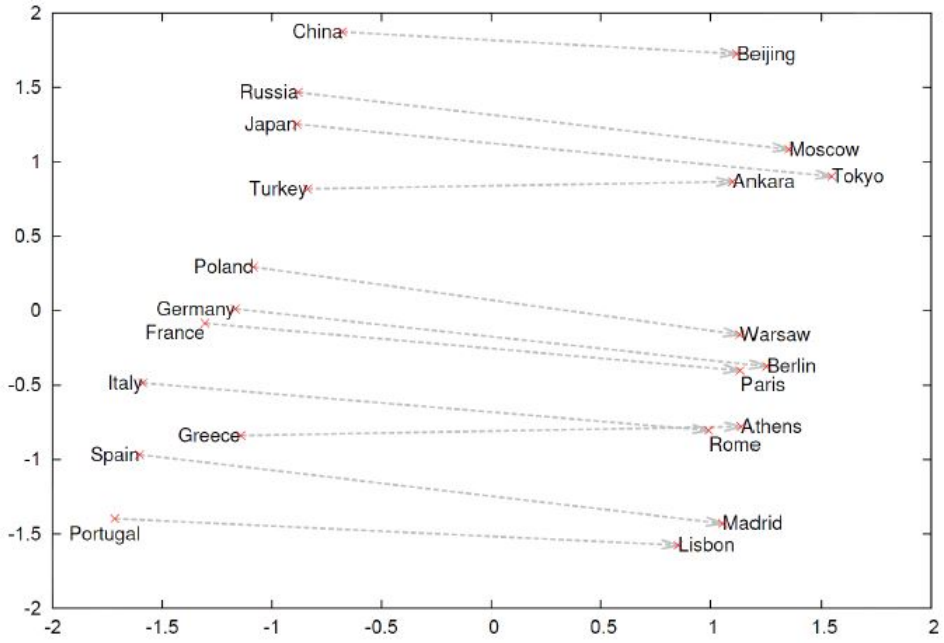


Figure 78.: Projection sur les deux composantes principales de l'ACP des représentations vectorielles CBOW des villes et pays, Mikolov et al.

Aujourd'hui, la représentation vectorielle de la langue reste un sujet très actif dans la recherche associée aux réseaux de neurones. Plus récemment les modèles d'attention ont permis d'effectuer une forte avancée en matière de transfert de connaissances [Devlin et al. 2018]. Jusqu'à 2018, les représentations vectorielles, malgré l'entraînement sur des données massives, possédaient en effet une faible capacité de généralisation. Les travaux de Devlin et al. (2018) fournissent ainsi une représentation permettant de répondre à de nombreux problèmes d'apprentissage statistique. Le paragraphe qui suit expose ainsi succintement le modèle BERT qui semble aujourd'hui devenir une référence dans la résolution de nombreux problèmes d'analyse sémantique.

4.3.3.3 *Zoom sur BERT [Devlin et al. 2018]*

Le modèle BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al. 2018] est un modèle *deep learning* qui définit un nouvel état de l'art pour 11 problèmes de NLP sans avoir à effectuer des ajustements spécifiques. Il se base sur 2 innovations :

- L'utilisation de l'architecture Transformer [Aswani 2018] pour réaliser un pré-entraînement bi-directionnel profond. BERT est composé d'une succession d'encodeurs (cf figure 79), 12 pour la version basique et 24 pour la version large.

- L'utilisation de deux nouvelles tâches pour le pré-entraînement : la prédiction des mots aléatoirement masqués (Figure 80) et la classification binaire de la phrase suivante, soit prédire pour 2 phrases A et B, si B est la phrase qui suit A.

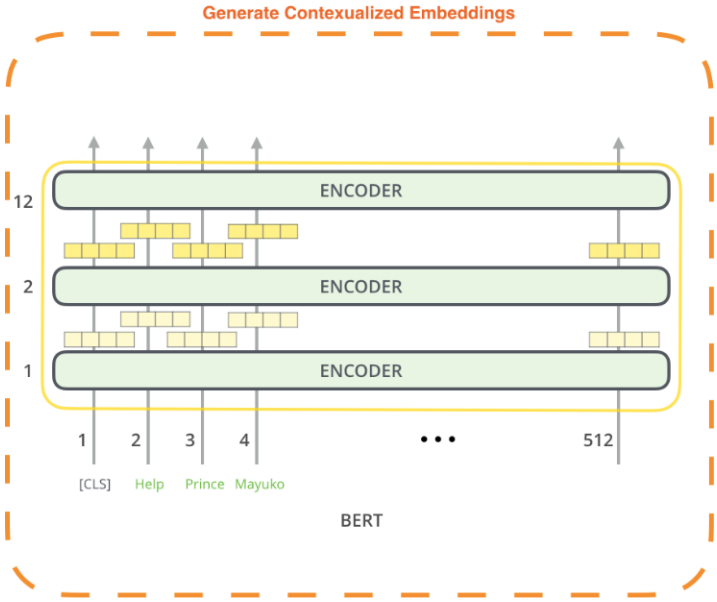


Figure 79.: Illustration de la structure de BERT. Source http://jalammar.github.io

Une fois pré-entraîné, il possède une représentation des mots qui prend en compte le contexte sémantique et syntaxique (soit une représentation linguistique) et permet ainsi de résoudre des problèmes de classifications supervisées.

La particularité de BERT est qu'il possède sa propre segmentation des mots. Pour toutes les langues –à l'exception du chinois– il utilise en effet des morceaux de mots (*wordpieces* [Yonghui Wu 2016]). Par exemple pour le mot 'unaffable' l'algorithme décompose en la séquence 'un', '##aff', '##able'. Le symbole '##' permet d'indiquer que le morceau n'est pas le début d'un mot.

Par ailleurs, en pratique, l'entrée du modèle BERT est particulière (Figure 80). Elle se compose d'une phrase segmentée qui commence par un marqueur spécial '[CLS]' et se termine par '[SEP]'. La phrase est ensuite transformée en 3 vecteurs :

- *Token Embeddings* est le vecteur de représentation des *wordpieces*. Chaque *wordpiece* est représenté par un vecteur de taille 768.
- *Sentence Embedding* est le vecteur qui permet de distinguer 2 phrases lorsqu'on met 2 phrases en l'entrée du modèle. La valeur 0 indique que le *wordpiece* appartient à la 1^e phrase et 1 pour la 2^e phrase.

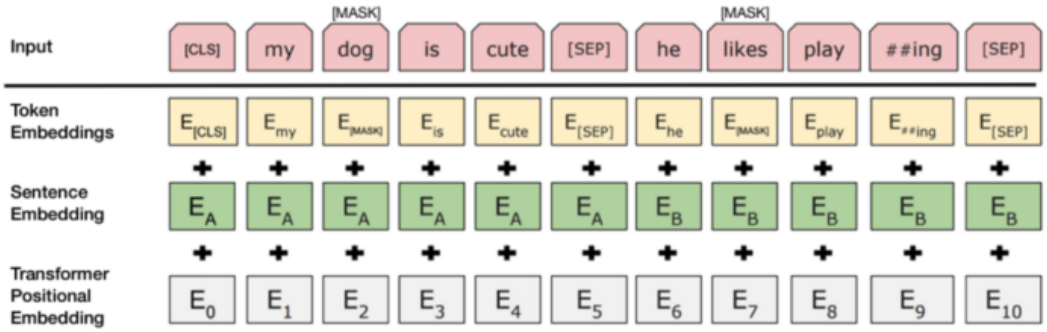


Figure 80.: Représentation de l'input de BERT et des mots masqués

- *Transformer Positional Embedding* est le vecteur qui contient l'information relative au positionnement des mots. Par exemple dans "I think, therefore I am", le 1^e I ne devrait pas avoir la même représentation vectorielle que le 2^e.

La sortie du modèle BERT peut être une liste de vecteurs pour chaque mot, ou bien un *sentence embedding* contenu dans le token '[CLS]'. Il est également possible d'ajouter une couche linéaire pour effectuer une classification. Aujourd'hui, cette nouvelle architecture constitue une nouvelle référence dans la littérature. Les performances présentées sur différentes tâches et une grande quantité de problèmes laissent penser que la représentation des mots que ce modèle permet est celle qui capture le plus précisément aujourd'hui les mécanismes de la langue.

Ainsi, le *deep learning* constitue une discipline incontournable dans l'extraction d'information à partir de documents. De l'OCR au traitement automatique du document, ces méthodes semblent fournir des performances non équivoques. Ce chapitre présente ainsi certaines méthodes avec lesquelles nous avons dû nous familiariser lors de la thèse afin de répondre à des besoins industriels. Les travaux présentés sont pour certains toujours en cours et font l'objet d'amélioration continue en fonction des différentes avancées de la recherche. Nous avons pu par ailleurs intégrer l'ensemble des méthodes introduites en dernière section dans une librairie que nous avons mis à disposition de l'entreprise. Ces travaux ont ainsi été l'occasion de faire face aux enjeux opérationnels d'utilisation de modèles complexes, de leur maintenance ainsi que leur déploiement.

5

IMPACTS DE L'ANONYMISATION DES DONNÉES SUR LES MODÈLES TARIFAIRES

Tout au long de la thèse, nous avons pu constater les rapides avancées sur les modèles neuronaux profonds [LeCun et al. 2015]. Leurs performances sur de nombreux sujets souvent liés au traitement de données non-structurées comme les images, le texte ou le son sont également dépendantes de l'accessibilité à des quantités de données significatives et des ressources de calcul toujours croissantes. La recherche frénétique sur ce sujet amène à devoir effectuer une veille permanente et une adoption de plus en plus challengée par l'intégration des librairies associées aux publications ainsi qu'à la transposition des algorithmes pré-calibrés sur les données des industriels [Torrey and Shavlik 2010]. Cet enjeu s'accompagne d'une volonté de tirer le meilleur des nouvelles technologies et d'une nécessité de transformation des industries. Depuis le début de la thèse, nous sommes témoins de cette transition : la donnée est désormais centrale dans la totalité du secteur tertiaire et plus particulièrement celui de l'assurance. Néanmoins, la collecte des données pose de nouvelles questions et notamment celle de la protection des utilisateurs du numérique.

Depuis mai 2018, le Règlement Général sur la Protection des Données (RGPD)[1] encadre l'utilisation des données à caractère privé concernant tout citoyen européen. Elle pose notamment un cadre strict sur l'enregistrement des données et les responsabilités des acteurs. En outre, l'utilisation des données doit s'accompagner d'une information rigoureuse auprès des propriétaires et leur garantir un droit de retrait s'ils manifestent cette volonté. Cette obligation impose ainsi des contraintes techniques et technologiques. Nous focalisant sur les assurances, nous avons donc étudié différentes méthodes permettant d'être conforme à cette réglementation [Ly and Poinsignon 2019]. Nous avons étudié différentes techniques permettant d'assurer la sécurité des données mais également leur anonymat. Ce chapitre se focalise principalement sur les impacts de l'anonymisation des données sur les processus de tarification automobile. La publication de l'article associé à ce chapitre et écrit avec Thomas Poinsignon et Romulad Elie est attendue pour le dernier trimestre 2019. Ces travaux ont par ailleurs été présentés lors du Congrès Européen des Actuaires en juin 2019 à Lisbonne.

1 https://www.cnil.fr/fr/reglement-europeen-protection-donnees

5.1 LA PROTECTION DES DONNÉES PERSONNELLES, UN ENJEU MAJEUR

5.1.1 *Le Réglement Général sur la Protection des Données (RGPD)*

Le positionnement des données au centre des stratégies dans l'ensemble des industries soulève rapidement la question de la protection des données personnelles. Afin de développer des services comme celui des recommandations commerciales ou de l'indexation des moteurs de recherche, les acteurs accumulent des informations sur leurs utilisateurs. Afin d'éviter toute dérive, l'Union Européenne a ainsi mis en place le Règlement Général sur la Protection des Données (RGPD). Applicable depuis le 25 mai 2018, ce dernier renforce et unifie la protection des informations personnelles des citoyens européens. En outre, il permet aux entités nationales de régulation, comme la Commission Nationale de l'Informatique et des Libertés (CNIL) en France, d'accroître la responsabilité des acteurs du numérique. Les assurances n'y échappent pas et doivent ainsi respecter le RGPD sous peine de sanctions lourdes allant jusqu'à 4% du chiffre d'affaire du groupe. Plus généralement le RGPD propose :

- *Un cadre harmonisé*, permettant une meilleure lisibilité législative au sein des pays de l'Union Européenne où désormais les mêmes règles s'appliquent.
- *Une application en dehors des frontières de l'UE*, notamment pour les entreprises non-européennes traitant des données issues de résidents ou d'organisations européennes.
- *Un consentement " explicite "* de la part des utilisateurs/clients quant à l'utilisation de leurs données personnelles ainsi que davantage de contrôle sur ces dernières.
- *Davantage de responsabilisation des acteurs et des sanctions plus importantes*, visant ainsi à réduire le nombre de formalités à fournir auprès du régulateur afin de faciliter les opérations des entreprises, tout en instaurant des pénalités renforcées en cas de non respect des règles pouvant s'élever jusqu'à 4% du chiffre d'affaires mondial annuel de la compagnie.
- *Un droit à l'effacement et à la portabilité des données personnelles*, permettant à chaque individu de demander l'effacement de données à caractère personnel dans les meilleurs délais ainsi que la possibilité d'obtenir ses données dans un format structuré et interprétable de manière automatique afin notamment de pouvoir les transmettre à un autre organisme de traitement.
- *La nomination d'un délégué à la protection des données*, obligatoire lorsque les opérations réalisées sur ces données nécessitent un suivi régulier à grande échelle des individus concernés. Il doit alors s'assurer du respect de la réglementation mais également conseiller les différents acteurs des traitements sur son application et ainsi devenir le principal interlocuteur de l'entreprise avec l'autorité de contrôle.

- *Des principes de " protection des données dès la conception " et de " sécurité par défaut "*, qui imposent aux entreprises de prendre en considération les enjeux liés à la protection des données à caractère privé dès la conception de leurs produits et services, et ce durant la totalité de son processus d'élaboration.

5.1.2 *La pseudonymisation et l'anonymisation*

Le RGPD conduit donc les entreprises qui manipulent des données personnelles à recourir à des moyens suffisants pour garantir la sécurité des informations à tout niveau : sur le stockage (éviter les fuites de données) et leur manipulation (avoir un cadre bien défini d'utilisation). Ces processus de sécurisation peuvent alors se décliner en différentes méthodes :

- La pseudonymisation des données : ces techniques permettent de manipuler des données personnelles sans révéler directement les informations qu'elles contiennent (par exemple en utilisant des chiffrements). Plus précisément, l'article 4 du RGPD introduit la *pseudonymisation* comme étant "*le traitement de données à caractère personnel de telle façon que celles-ci ne puissent plus être attribuées à une personne concernée précise sans avoir recours à des informations supplémentaires, pour autant que ces informations supplémentaires soient conservées séparément et soumises à des mesures techniques et organisationnelles afin de garantir que les données à caractère personnel ne sont pas attribuées à une personne physique identifiée ou identifiable.*"

- L'anonymisation des données : ces techniques permettent de préserver de manière définitive l'anonymat d'une information. Elles doivent donc pouvoir être robustes à toutes méthodes statistiques ou algorithmiques qui tenteraient de pouvoir associer une information à une personne physique.

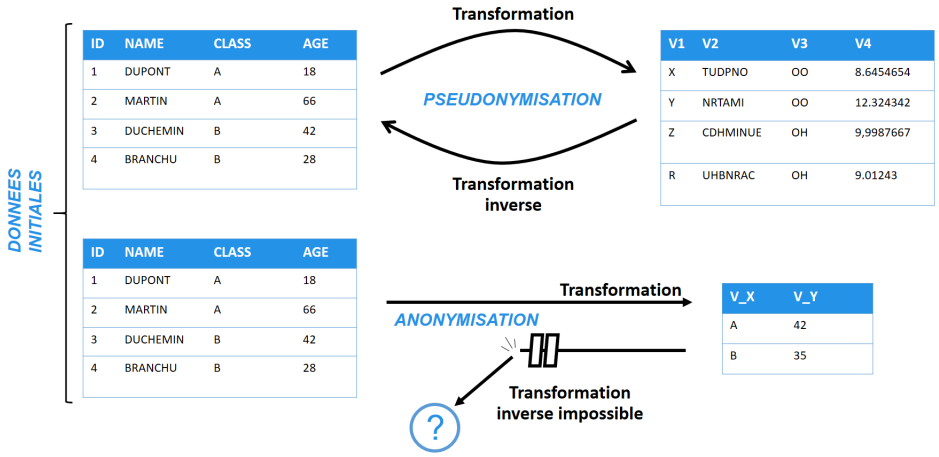


Figure 81.: Différences entre anonymisation et pseudonymisation

Dans le cadre de la pseudonymisation, différentes techniques permettent par exemple de créer des tables de correspondance entre les informations personnelles sensibles (telles que le nom, prénom, adresse, etc.) et une clé obtenues souvent à l'aide d'une fonction de Hash [Damgård 1989]. Dowlin et al. (2016) proposent également une méthode qui consiste à chiffrer intégralement le contenu d'une donnée tout en permettant à un tiers d'effectuer des calculs. Ces techniques permettent ainsi de sécuriser les données lorsqu'elles sont transmises à un prestataire. Il devient alors impossible pour le manipulateur de données (statisticien, actuaire, etc.) d'identifier le contenu tout en effectuant des opérations mathématiques [Aslett et al. 2015]. Aujourd'hui, les additions et multiplications sont le plus souvent préservées et permettent un ensemble limité d'opérations. Nos expérimentations menées avec Poinsignon (2018) confirment alors qu'aujourd'hui ces méthodes de chiffrement intégral de la donnée ne sont pas industrialisables.

Par ailleurs, le RGPD impose un cadre strict dès lors qu'une donnée contient des informations à caractère personnel. En outre, l'article 25 du RGPD prévoit que "*le responsable du traitement met en œuvre [...] des mesures techniques et organisationnelles appropriées, telles que la pseudonymisation, qui sont destinées à mettre en œuvre les principes relatifs à la protection des données*". Si la pseudonymisation est une protection minimale des données, les tables contenant des identifiants permettant de faire le lien avec un individu sont alors soumises aux conditions de la réglementation : droit d'utilisation de la donnée restreint dans le temps et à une liste de tâche bien précise, obligation de moyens en terme d'auditabilité de la donnée et des personnes qui y ont accès, etc.

L'anonymisation consitue la meilleure protection des données personnelles. De ce fait, elle permet de ne pas être soumis à de nombreuses contraintes du RGPD. Néanmoins,

dans un cadre de modélisation du risque, la maille individuelle permet de mieux comprendre les différents facteurs influençant la probabilité d'un sinitre et donc effectuer une meilleure mutualisation. Il ainsi important de ne pas dégrader la performance des modèles potentiellement calibrés sur des données individuelles pseudonymisées. C'est pourquoi nous nous sommes intéressés à l'impact des méthodes d'anonymisation sur certains modèles actuariels et plus spécifiquement les modèles de tarification non-vie d'assurance automobile.

5.2 L'ANONYMISATION DES DONNÉES

5.2.1 *Etat des lieux des méthodes d'anonymisation*

La littérature propose différentes techniques d'anonymisation de la donnée [Nguyen 2014]. Nous introduisons dans ce paragraphe trois grandes familles de techniques explorées dans [Ly and Poinsignon 2019]. La première consiste à chiffrer la donnée [Blanc and A. 2004] et à ensuite effacer la clé de cryptage empêchant ainsi tout exploitation des données personnelles. Si cette méthode peut laisser -selon certaines conditions- la possibilité d'effectuer des opérations sur la donnée [Yi 2014], elle rend néanmoins impossible toute interprétation du modèle. Elle ne présente donc que peu d'intérêt.

Une seconde famille de méthodes consiste à transformer la donnée pseudonyme en y introduisant du bruit. Néanmoins, ces techniques n'empêchent pas de pouvoir ré-identifier les individus. Même si les informations personnelles sont masquées par pseudonymisation et les autres variables bruitées, certaines statistiques telles que la moyenne ou la variance restent souvent préservées par ces méthodes. Par ailleurs Narayanan and Shmatikov (2008) ont démontré qu'il était possible même si aucune information à caractère personnel n'est mise à disposition, de tout de même pouvoir ré-identifier un individu. Si les différentes variables exogènes attribuées à un individu sont nombreuses, il devient alors possible d'isoler ce dernier. Le RGPD considère alors ces familles de méthodes comme non suffisantes pour considérer la donnée comme réellement anonyme.

Enfin, une troisième méthode consiste à utiliser un principe de généralisation. Cette approche consiste "à diluer (ou généraliser), les attributs des personnes concernées en modifiant leur échelle ou leur ordre de grandeur respectif (par exemple, une région plutôt qu'une ville, un mois plutôt qu'une semaine). Si la généralisation peut être efficace pour empêcher l'individualisation, elle ne garantit pas une anonymisation effective à 100% et doit donc être combinée avec d'autres techniques"[2]. Parmi ces techniques, il est intéressant de mentionner la *k-anonymisation* [Sweeney 2002] et la *I-diversité* [Machanavajjhala et al. 2006]. Ces deux méthodes sont en effet souvent combinées afin de garantir l'anonymat des données.

2 D'après le RGPD

5.2.2 *La k-anonymisation*

Ainsi, la *k-anonymisation* permet d'empêcher d'identifier un individu au sein d'une base de données en regroupant k enregistrements sous une même modalité. La Figure 82 ci-dessous illustre la procédure de *k-anonymisation* en prenant $k = 3$ sur une base de données où l'on considère la variable `Damage` comme une donnée sensible. L'intérêt de la *k-anonymisation* réside ainsi dans sa capacité à empêcher l'identification d'un individu par recoupement des informations avec d'autres bases de données. De plus, cette procédure permet de préserver de nombreuses statistiques et ne bruite pas l'information. Néanmoins, les k enregistrements sont systématiquement affectés à un même groupe. Cette méthode, utilisée seule, connaît plusieurs vulnérabilités, notamment aux attaques par homogénéité (a) ou par connaissance antérieure (b) :

a. La Figure 82 représente une base 3-anonyme vulnérable à une attaque par homogénéité. Lorsque les enregistrements d'un même groupe possèdent la même valeur de la variable sensible (celle dont on souhaite spécifiquement protéger l'accès), les données ne sont alors plus anonymisées. Sur le schéma, on peut par exemple déduire qu'une personne exerçant une profession libérale, âgée de 39 ans a nécessairement subit un sinistre très important.

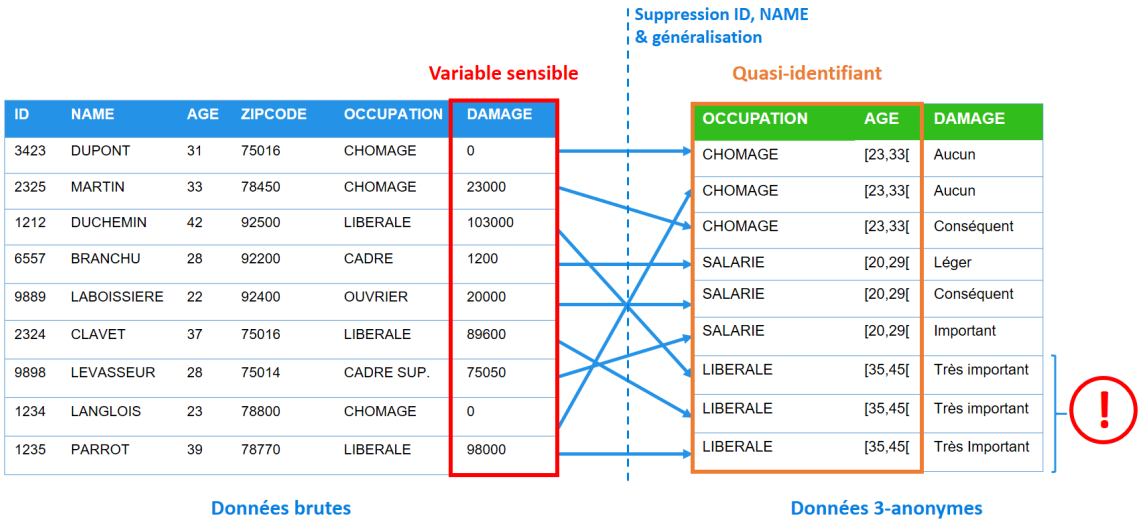


Figure 82.: Schéma & limites de la k-anonymisation

b. Par ailleurs, si un attaquant connaît suffisament d'informations sur un individu, alors il peut inférer de la base k-anonymisée des informations sensibles. Par exemple à partir du schéma (82), si M. DUMONT sait que son voisin (M. MARTIN) est

au chômage alors qu'il vient de fêter son 33^{eme} anniversaire et qu'il a récemment eu un accident, alors il peut en déduire qu'il s'agissait d'un sinistre conséquent.

Par ailleurs, ces méthodes possèdent également quelques limitations techniques dans la détermination des quasi-identifiant optimaux qui nécessite d'utiliser des heuristiques comme celles proposées par Bayardo and Agrawal (2005).

5.2.3 *La I-diversité*

La procédure de *I-diversité* [Machanavajjhala et al. 2006] est complémentaire à l' *k-anonymisation* et permet de limiter les risques d'attaque par homogénéité. Cette méthode impose notamment une contrainte supplémentaire à la *k-anonymisation* : chaque groupe construit doit contenir un minimum de I modalités différentes. Cette contrainte modifie alors la construction des quasi-identifiants. La Figure 83 illustre ainsi le résultat sur l'exemple exposé précédemment.

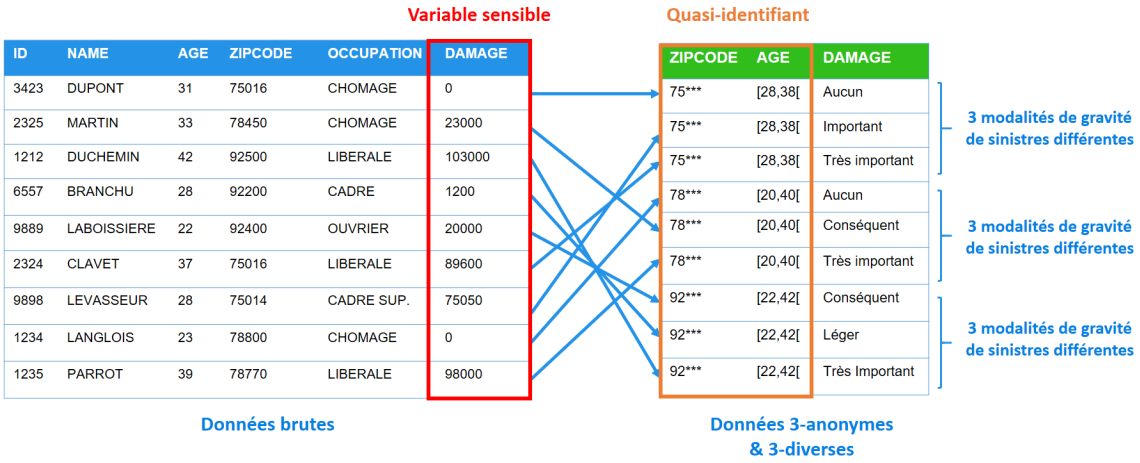


Figure 83.: Schéma & limites de la I-diversité

Néanmoins cette procédure présente encore certaines limitations. En effet si les modalités de la variable sensible ne sont pas convenablement réparties au sein des différentes classes alors il est possible d'inférer certaines informations. Par exemple dans la Figure 83, on peut être certains que toutes les personnes habitant dans le département 92 ont subit un sinistre. Mais comme rappelé par Ly and Poinsignon (2019), la répartition au sein d'une même classe des modalités de la variable sensible a également son importance. En effet en considérant une variable sensible à seulement deux modalités (booléenne) dans le cadre d'une base 100-anonymisée, si une des classes de 100 enregistrements comporte 99 fois la valeur 1 et une fois la valeur 0 pour la variable sensible, alors si un

attaquant connaît le quasi-identifiant d'un individu appartenant à cette classe il peut en déduire avec une grande probabilité que la variable sensible pour cet individu vaut 0. La méthode de t-proximité [Li et al. 2007] propose alors une solution à cette limite mais décorrèle fortement les variables entre elles, rendant ainsi toute tentative de modélisation caduque (étant donné que l'on tente d'expliquer le plus souvent une de ces variables en fonction des autres).

Ces méthodes sont par ailleurs limitées également par la dimension des bases de données [Aggarwal 2005]. En effet, par simple effet de combinatoire, plus le nombre de variables et de modalités qu'elles contiennent devient grand, plus il devient facile de pouvoir isoler un individu. Ainsi, appliquer strictement la combinaison de *k-anonymisation* et de *I-diversité* afin de préserver l'anonymat engendrerait une perte d'information conséquente. Toutes ces restrictions nous ont donc amené à s'inspirer de ces familles de méthodes et à explorer d'autres alternatives de généralisation.

5.3 ETUDE D'IMPACTS SUR LES MODÈLES DE TARIFICATION AUTOMOBILE ACTUELS

5.3.1 *Cadre de notre étude*

Notre étude s'inscrit dans un cadre bien précis de management de la donnée d'assurés afin d'être en mesure d'effectuer une tarification automobile. Nous considérons ainsi une compagnie d'assurance souhaitant préserver strictement l'anonymat (au sens de RGPD) de ses assurés lors de l'élaboration de son modèle tarifaire. Ainsi, dans notre cadre, les données sont exposées à soit un risque de fuite via une attaque du réseau par exemple, soit un risque de manipulation malveillante par une personne interne à la compagnie (qui pourrait de surcroît, avoir accès lors de la modélisation aux données individuelles). Par la suite, nous ne considérons donc pas la possibilité de requêtes sur la base de données autres que celles de la part d'un employé de la compagnie. L'information est alors entièrement disponible uniquement en interne et nous supposons que l'entreprise est en milieu fermé (aucune information ne peut être recoupée avec d'autres données que celles dont la compagnie dispose). Notre objectif est ainsi d'étudier la possibilité d'anonymisation des données utilisées à des fins de tarification. Ce qui sera présenté par la suite pourra par ailleurs être vu également comme une méthode d'analyse applicable à n'importe quelles données de tarification assurantiel. Ces données sont en effet aujourd'hui souvent pseudonymisées et éventuellement *k-anonymisées*.

S'inscrivant dans la même philosophie que les méthodes par agrégation, nous étudions l'utilisation des méthodes d'apprentissage statistiques non-supervisées à des fins d'anonymisation et étudions leurs impacts sur les précisions d'un modèle tarifaire calibré

sur de telles données. L'utilisation des méthodes de classification non-supervisées semble en effet présenter deux avantages majeurs :

- Elles permettent de garantir que les données sont anonymisées en proposant l'élaboration d'individus représentatifs d'une classe. Il devient alors impossible de retrouver l'information provenant d'un individu spécifique une fois l'agrégation réalisée. Nous détaillons par la suite, les contraintes que nous utilisons pour fournir de telles garanties.

- Elles constituent de surcroît un bon moyen de compresser la base de données initiale en perdant le moins d'informations possible pouvant être utile à l'élaboration d'un tarif, là où les méthodes précédemment mentionnées s'avèrent loin d'être optimales. Elles permettent pour certaines de proposer également un modèle génératif de données individuelles. Nous étudions l'impact de cette compression d'information sur les modèles.

5.3.1.1 *Méthodologie générale*

Notre étude se focalise sur un modèle de tarification automobile pour des types de garanties spécifiques (matérielle et corporelle). Nous discutons ce point en conclusion de ce chapitre. Par soucis de comparaison, notre étude d'impacts se fera au regard d'un modèle de tarification calibré selon les standards de marché[3]. Ce modèle de référence permettra ainsi de comparer à iso-méthologie l'impact de l'utilisation des données agrégées sur le calcul de la prime pure. Nous appliquons donc à chaque fois rigoureusement la même méthode de calibrage (i.e. le même modèle de prime pure) mais sur des données respectivement pseudonymes et anonymes.

Notre étude d'impact suit alors les étapes suivantes :

- Calibrage du modèle de référence sur 60% de la base de données initiale (telle qu'un assureur pourrait utiliser actuellement). Il s'agit des données des assurés sur potentiellement plusieurs années d'observation. On appellera ce modèle, le modèle "ligne à ligne".

- Calibrage de la même structure de modèle mais cette fois-ce sur l'anonymisation par agrégation des mêmes 60% de la base de données initiale. On utilise ainsi initialement la même base d'apprentissage mais "résumée" par notre méthode d'apprentissage non-supervisé. Elle contient donc nécessairement moins de ligne que celle utilisée pour calibrer le modèle de référence. On appellera ce modèle, le modèle "anonymisé".

3 Nous ne discutons pas ici les méthodes de tarification. Ainsi nous nous focalisons sur les résultats initiaux de primes pures mais détaillons la méthodologie mise en place pour les obtenir. Cette dernière s'appuie sur l'expertise de différents consultants Milliman.

- Comparaison des résultats de ces deux modèles sur les 40% de la base restants où chaque ligne représente un assuré. Les 40% ne sont donc pas anonymisés et représentent par exemple de nouveaux assurés entrant dans le portefeuille.

Les ratio 60% et 40% sont discutés par la suite. Ainsi, à une méthodologie de tarification donnée, nous étudions l'impact sur le calcul de la prime pure des méthodes d'anonymisation utilisés. Ces impacts seront étudiés sur la prime pure moyenne (moindre carrés, statistiques agrégées, etc.) mais également sur les erreurs individuelles engendrées par l'anonymisation (distribution des écarts relatifs).

On comprend alors que l'enjeu principal pour réaliser notre modèle anonymisé réside dans le choix de la méthode de partitionnement des données afin que celui-ci permette d'obtenir des groupes d'individus dont leurs caractéristiques (les "X") soient les plus homogènes possibles (sans pour autant avoir de certitude sur l'homogénéité de leurs sinistralités respectives).

Les données utilisées (issues du *Pricing Game*[4] 2015) sont détaillées en Annexe page 305. La base contient en outre l'information de 100 000 polices d'assurance automobile observées sur 2 ans. La base après retraitement contient 100 000 lignes et 20 variables parmi lesquelles nous retenons que 12 variables pour la construction de notre modèle.

Dans les paragraphes qui suivent, nous étudions donc le comportement des modèles de primes pures calibrés sur ces données.

5.3.1.2 *Précisions sur la méthodologie de tarification*

Le modèle de tarification retenu utilise une approche fréquence/sévérité. C'est-à-dire que les fréquences de sinistres (à la fois matériels et corporels) et leurs coûts individuels moyens sont modélisés indépendemment. En posant Y_1 = Fréq. matérielles, Y_2 = Fréq. corporelles, Y_3 = Coût Moy. matériels et Y_4 = Coût Moy. corporels, on souhaite ainsi pouvoir prédire le montant de la prime pure annuelle à payer pour un assuré comme étant l'espérance de sa charge sinistre, soit avec les notations introduites :

$$\text{Prime Pure}_{\text{pred}} = \mathbb{E}[Y_1.Y_3 + Y_2.Y_4]$$

Afin de tenir compte de l'exposition de chaque police au sein de notre portefeuille, nous normalisons les fréquences corporelles et matérielles par la durée de prise en charge de l'assuré par l'assureur (avec la variable `EXPOSITION`, cf annexe A.5).

Pour ce faire nous introduisons les variables :

- `WEIGHT` $= \frac{\texttt{EXPOSITION}}{365}$, utilisée pour normaliser les variables en fonction de l'exposition de la police dans le portefeuille.

4 www.freakonometrics.com

name	simple_type	nb_modalities	max	mean	min	stddev
POL_NUM	numeric	100000	200285805	200200338	200114871	62166.635234436595
CAL_YEAR	numeric	2	2016	2015.5	2015	0.5000025000138331
EXPOSITION	numeric	275	365	327.58025	91	73.5704059115063
DRIVER_GENDER	string	2	Male	-	Female	-
DRIVER_OCC	string	5	Unemployed	-	Employed	-
DRIVER_AGE	numeric	58	75	41.12506	18	14.298972851315035
VEH_TYPE	string	6	F	-	A	-
VEH_CATEGORY	string	3	Small	-	Large	-
VEH_GROUP	numeric	20	20	10.69284	1	4.6873737262058155
VEH_VALUE	numeric	9395	49995	16454.6194	1000	10507.019806457773
POL_DURATION	numeric	16	15	5.47093	0	4.591155162065123
BONUS_MALUS	numeric	21	150	-6.93	-50	48.62794203713991
DAMAGE_GUAR	numeric	2	1	0.5122	0	0.4998536371144925
COUNTY	string	471	U9	-	L1	-
STATE	string	10	U	-	L	-
COUNTY_DENSITY	numeric	471	297.3851697	117.15688049021784	14.37714238	79.49898804465151
NUM_LIAB_DAM	numeric	8	7	0.14724	0	0.4366947325339446
NUM_LIAB_BOD	numeric	4	3	0.04678	0	0.2195270223028032
INC_LIAB_DAM	numeric	12257	12878.36991	106.15729505072366	0	444.9932471326946
INC_LIAB_BOD	numeric	4505	69068.02629	222.7432719670955	0	1859.5625777467174

Figure 84.: Récapitulatif des variables du jeu de données

- $\texttt{MEAN\_INC\_BOD} = \begin{cases} \frac{\texttt{INC\_LIAB\_BOD}}{\texttt{NUM\_LIAB\_BOD}} & \text{Si } \texttt{NUM\_LIAB\_BOD} > 0 \\ 0 & \text{Sinon} \end{cases}$, coût moyen des sinistres corporels.
- $\texttt{MEAN\_INC\_DAM} = \begin{cases} \frac{\texttt{INC\_LIAB\_DAM}}{\texttt{NUM\_LIAB\_DAM}} & \text{Si } \texttt{NUM\_LIAB\_DAM} > 0 \\ 0 & \text{Sinon} \end{cases}$, coût moyen des sinistres matériels.

On considére par la suite $Y_1 = \texttt{NUM\_LIAB\_DAM}$, $Y_2 = \texttt{NUM\_LIAB\_BOD}$, $Y_3 = \texttt{MEAN\_INC\_DAM}$ et $Y_4 = \texttt{MEAN\_INC\_BOD}$.

La famille des modèles utilisée pour la fréquence et le coût moyen est celle des modèles linéaires généralisés (GLM) que nous détaillons en annexe A.6 page 316. Cette famille de modèles est aujourd'hui la plus utilisée pour la tarification des primes pures associées aux sinistres matériels et corporels. Nous renvoyons également le lecteur à l'annexe pour plus de détails sur les retraitements appliqués à la donnée et les choix des distributions

associés aux modèles. A titre indicatif, les performances générales du modèle de référence calibré sur 60% de la donnée initiale sont présentées ci-après.

Table 22.: Résultats de la méthode GLM "ligne à ligne" sur les 40% de données de test (modèle de référence calibré sur les données non-anonymisées)

$\sum$ tot. prédictions	12 662 065.20€
$\sum$ tot. observées	12 580 726.73€
Ecart (predictions - observées)	-81 338.48€
Ecart relatif	0.66%
RMSD[5]	1 863,19€

5.3.2 *Construction des données anonymisées et impacts*

On rappelle que l'objectif de cette étude est de mesurer l'impact sur les modèles tarifaires des méthodes d'anonymisation par agrégation qui s'appuient sur l'apprentissage statistique non-supervisé .

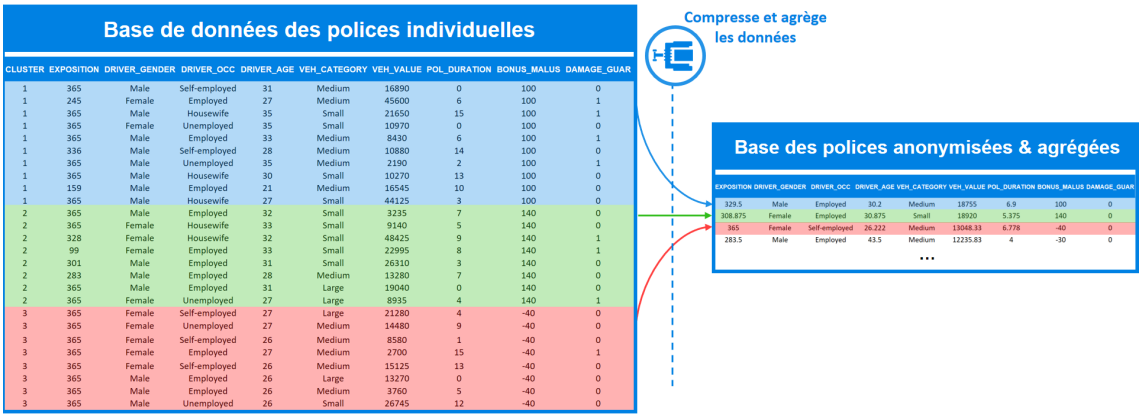


Figure 85.: Schéma de l'agrégation de polices d'assurances dans un contexte d'anonymisation

5 Root-mean-square deviation, i.e : RMSD $= \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$

Nous nous sommes donc intéressés à différents algorithmes de classification non-supervisés et plus particulièrement :

- K-means [Lloyd 1982] : cette méthode consiste, pour un entier $K \in \mathbb{N}^*$ donné, à constituer K groupes les plus homogènes possibles par minimisation de la variance intra-classe et la maximisation de la distance entre les classes. La résolution de cet algorithme de maximisation d'espérance (EM) peut alors être approchée par une procédure que nous détaillons ci-après.

- Classification Ascendante Hiérarchique (CAH) [Ward Jr 1963] : cette méthode (détaillée en annexe A.7) consiste à constituer des groupes d'observations de manière hiérarchique en s'appuyant sur une mesure de similarité. Ainsi, l'ensemble des observations sont regroupées de manière itérative deux à deux. Puis chaque groupe ainsi constitué est de nouveau regroupé deux à deux jusqu'à l'obtention d'un segment unique contenant toute la donnée. A la fin de l'algorithme, l'utilisateur peut ainsi choisir l'itération de regroupement (i.e. la hiérarchie ou le niveau d'agrégation) qui convient le mieux.

- OPTICS [Ankerst et al. 1999] : cette méthode est une variante de l'algorithme DBSCAN [Ester et al. 1996]. Elle consiste en une méthode itérative permettant de rassembler des points connectés par densité. Nous détaillons ci-après cette méthode utilisée le plus souvent dans la détection d'anomalies.

- Propagation d'affinité [Dueck and Frey 2007] : cette méthode est similaire dans sa construction aux K-means à la différence où le nombre de classes est déterminé par l'algorithme. La propagation d'affinité permet en outre de déterminer les points les plus représentatifs de la base de données.

Ces quatre familles d'algorithmes ont donc été explorées afin de construire une base de données anonyme permettant d'impacter le moins possible notre modèle de tarification initial. Ainsi, dans l'application de ces algorithmes de classification non-supervisée, nous avons tenté d'agréger les observations de façon à :

- Privilégier la minimisation de l'hétérogénéité au sein des classes (i.e. la variance intra-classe) au détriment de la variance inter-classe faible. Ainsi, on souhaite préserver une certaine I-diversité et donc on pourra à ce titre imposer des contraintes supplémentaires sur les convergences des algorithmes de classification.

- Avoir suffisamment de groupes à l'issue de la procédure. On souhaite en effet conserver la diversité des risques représentée par les nouvelles observations représentant chaque groupe. Chaque agrégat représente en effet dans la base anonyme un individu fictif avec des caractéristiques représentatives d'un risque.

- Conserver des clusters pertinents et cohérents mais également avec un nombre d'individus suffisant afin de garantir l'anonymat des données (un minimum de deux observations par groupe est nécessaire pour assurer l'anonymat).

Les différentes paramétrisations des algorithmes se sont donc faites au regard de ces critères illustrés en Figure 86. Par ailleurs, seules les variables les plus discriminantes (à l'issue d'une analyse descriptive) et ne comptant pas de trop nombreuses modalités (comme `COUNTY, STATE`, etc.) ont été utilisées. La création des groupes représentatifs s'appuie alors sur variables suivantes :

- `DRIVER_AGE`
- `BONUS_MALUS`
- `COUNTY_DENSITY`
- `DRIVER_GENDER`
- `DRIVER_OCC`
- `VEH_CATEGORY`
- `DAMAGE_GUAR`

Une fois nos agrégats construits, les observations représentatives sont alors reconstruites à partir de l'intégralité des variables. Par exemple, au sein d'un même groupe, la modalité de la variable `COUNTY` par exemple est choisi comme le barycentre géographique des autres modalités présentes dans un même groupe. Si la pertinence de cette modalité peut être discutable elle permet néanmoins de garantir l'anonymat de la donnée.

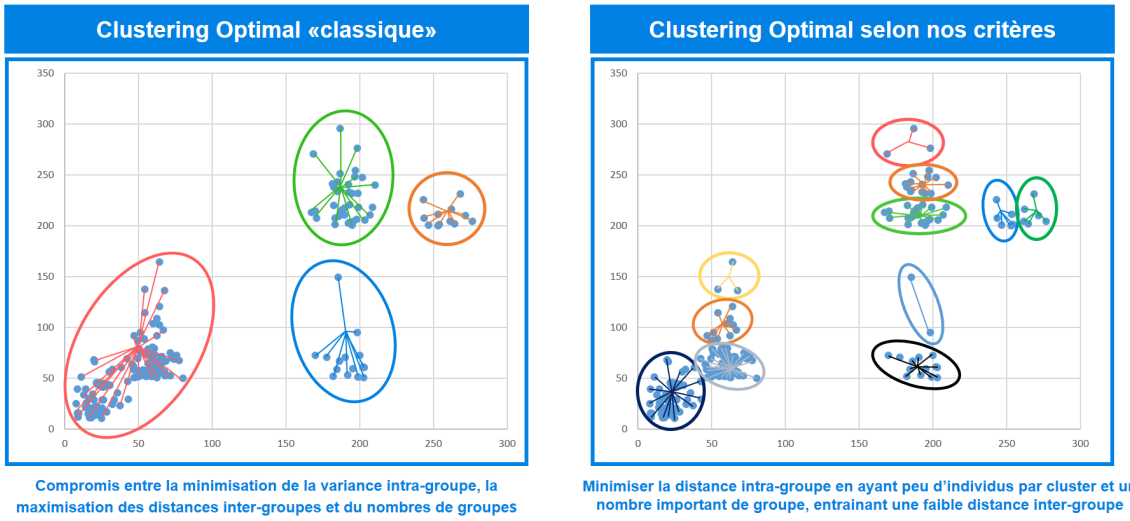


Figure 86.: Schéma des différences de critères de sélection des modèles de clustering

Nous détaillons dans les paragraphes suivants, les deux algorithmes qui nous ont apporté les résultats les plus intéressants. Le fonctionnement de ces derniers nous permettra notamment de mieux discuter les impacts obtenus sur le modèle de tarification.

5.3.2.1 *Zoom sur l'algorithme K-means*

Cet algorithme se définit au travers d'un problème de maximisation d'espérance ou *Expectation-Maximization* (EM) en anglais. Formellement, soit un ensemble de n individus $(x_1, x_2, \ldots, x_n)$ que l'on souhaite partitionner en K classes de C, tel que $C = \{C_1, C_2, \ldots, C_K\}$ où $K \leqslant n$, on cherche à minimiser la distance des points d'une classe k au barycentre du cluster μ_k (soit la variance intra-cluster) :

$$\underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i=1}^{n_k} (x_i - \mu_k)^2 \Leftrightarrow \underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \frac{1}{n_k} \operatorname{Var}(C_k)$$

Où $\mu_k := \overline{X_k} = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i$ avec n_k le nombre d'observations dans le cluster k. Par ailleurs grâce au théorème de König-Huygens on peut établir la relation suivante :

$$\sum \text{DIST. CARRES TOT.} = \sum \text{DIST. CARRES INTRA-CLUSTERS} + \sum \text{DIST. CARRES INTER-CLUSTERS}$$

Démonstration :

Nous commençons par définir l'indicatrice suivante :

$$\mathbb{1}_{x_i \in C_k} = \begin{cases} 1 \text{ Si } x_i \in C_k \\ 0 \text{ Sinon} \end{cases}$$

En posant μ, la moyenne de l'ensemble des observations (barycentre total) et en considérant donc que l'on a pour k fixé $\sum_{i=1}^{n} \mathbb{1}_{x_i \in C_k} = n_k$, on peut alors décomposer la somme des distances au carrés totale tel que :

$$
\begin{aligned}
\underbrace{\sum_{i=1}^{n}(x_i-\mu)^2}_{\text{Dist. totale}} &= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu)^2 \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu_k+\mu_k-\mu)^2 \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}\left[(x_i-\mu_k)^2+2(x_i-\mu_k)(\mu_k-\mu)+(\mu_k-\mu)^2\right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu_k)^2 + 2\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu_k)(\mu_k-\mu) \\
&\qquad + \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(\mu_k-\mu)^2 \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu_k)^2 + 2\sum_{k=1}^{K}(\mu_k-\mu)\left[\sum_{i=1}^{n}\mathbb{1}_{x_i \in C_k}x_i - \sum_{i=1}^{n}\mathbb{1}_{x_i \in C_k}\mu_k)\right] \\
&\qquad + \sum_{k=1}^{K} n_k(\mu_k-\mu)^2 \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu_k)^2 + 2\sum_{k=1}^{K}(\mu_k-\mu)(n_k\mu_k-n_k\mu_k) \\
&\qquad + \sum_{k=1}^{K} n_k(\mu_k-\mu)^2 \\
&= \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{x_i \in C_k}(x_i-\mu_k)^2}_{\text{Dist. intra-clusters}} + \underbrace{\sum_{k=1}^{K} n_k(\mu_k-\mu)^2}_{\text{Dist. inter-clusters}}
\end{aligned}
\tag{29}
$$

□

En remarquant que la somme des distances au carré (variance totale) est constante quelle que soit la valeur de K, l'algorithme du K-means est donc équivalent à maximiser la variance inter-clusters (l̃a distance entre les classes) [6].

6 En effet on a : $\underset{intra}{\operatorname{argmin}}\, intra = Cst - inter \Leftrightarrow \underset{inter}{\operatorname{argmax}}\, inter = Cst - intra$

La spécification de K permet d'éviter l'obtention d'un résultat trivial où $K = n$, c'est à dire que chaque observation appartiendrait à son propre cluster, constitué uniquement de cette dernière. Dans ce cas la variance intra-clusters serait nulle (et donc minimale) et la variance inter-clusters maximisée, égale à la variance totale.

La résolution de l'algorithme dans sa formulation initiale est néanmoins NP-complexe. L'algorithme de Lloyd [Lloyd 1982] est alors couramment employé bien qu'il ne garantisse pas l'optimalité. Il permet néanmoins de converger rapidement en calculant successivement des extrema locaux qui dans certaines situations sont également globaux.

Algorithme de Lloyd

1. Sélectionner aléatoirement K points parmi les n points totaux pour devenir les "barycentres" initiaux des K clusters $(\mu_1^{(0)}, \mu_2^{(0)}, \ldots, \mu_K^{(0)})$

2. Tant qu'il n'y a pas convergence, à l'itération t :

   2.1 Classer chaque individu au sein du cluster dont le barycentre est le plus proche

   $$C_k^{(t)} = \left\{ x_i, \|x_i - \mu_k^{(t)}\| \leqslant \|x_i - \mu_{k^*}^{(t)}\| \forall k^* = 1, \ldots, K \right\}$$

   2.2 Recalculer le nouveau barycentre de chaque cluster

   $$\mu_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{x_i \in C_k^{(t)}} x_i$$

On notera que le critère de convergence peut être défini soit comme un nombre d'itérations maximal, soit comme une valeur de tolérance sur l'évolution des barycentres calculés tels que $\forall k$, $\|\mu_k^{(t)} - \mu_k^{(t+1)}\| \leqslant \epsilon$, avec $\epsilon > 0$.

L'utilisation de cet algorithme nous a ainsi permis d'étudier plusieurs agrégations au regard des différents points d'attention mentionnés dans la section précédente. Le tableau ci-dessous résume les différentes partitions réalisées (les impacts sur le modèle de tarification sont synthétisés en fin de cette section).

Table 23.: Les différents partitionnements réalisés à l'aide du K-means

Partitionnement	Nb. clusters base tot.	Nb. clusters base app. (60%)
KM_6000	10 000	6 000
KM_4500	7 500	4 500
KM_3000	5 000	3 000
KM_1200	2 000	1 200

5.3.2.2 *Zoom sur l'algorithme OPTICS*

L'algorithme OPTICS [Ankerst et al. 1999] est une extension de DBSCAN proposée par Ester et al. (1996). Afin de mieux comprendre son fonctionnement il convient de définir les notions suivantes.

Définition 5.1. On définit l'*epsilon-voisinage* $N_\epsilon(x)$ d'un point x quelconque du jeu de données C comme étant l'ensemble des observations de C dont la distance par rapport à x est strictement inférieure à ϵ, soit :

$$N_\epsilon(x) = \{u \in C,\, d(x,u) < \epsilon\} \qquad \forall x \in C$$

où d représente une mesure de distance sur C, tel que (C,d) définit un espace métrique. D'un point de vue topologique, on peut donc considérer $N_\epsilon(x)$ comme un voisinage de x dans C équivalent à une boule ouverte $\mathcal{B}(x,\epsilon)$ centrée en x de rayon ϵ.

Définition 5.2. On définit x comme un point intérieur si son espilon-voisinage contient au moins MinPoints tel que :

$$|N_\epsilon(x)| \geqslant \text{MinPoints}$$

Définition 5.3. On dit que les points x et u sont connectés par densité dans C s'il existe un ensemble $I := \{\nu_1, \ldots, \nu_m\}$ de m points intérieurs tel que :

$$x \in \Omega := \bigcup_{i=1}^{m} N_\epsilon(\nu_i)$$

où $\forall i \in \{2, \ldots, m\}$, $\nu_i \in N_\epsilon(\nu_{i-1})$ et $\nu_1 \in N_\epsilon(u)$.

L'algorithme DBSCAN surlequel s'appuie OPTICS propose alors une méthode non-supervisée afin de regrouper les points connectés par densité selon deux contraintes ϵ et $MinPoints$ définis plus haut. Le nombre d'agrégats obtenus est donc entièrement déterminé pas ces deux paramètres. OPTICS permet de s'abstenir du paramétrage ϵ en considérant ce paramètre comme infini par défaut. OPTICS se base alors sur deux

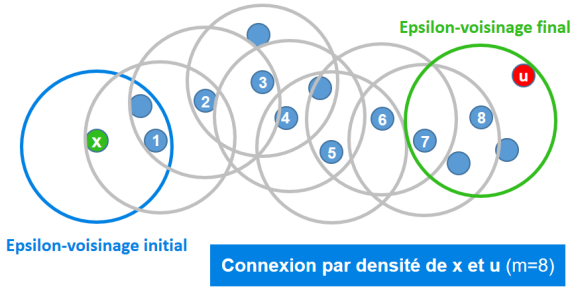


Figure 87.: Illustration de la connexion par densité de x et u (où MinPoints = 2)

nouvelles mesures associées à chaque observation. Tout d'abord, la distance interne (*core-distance*) d'un point u définie comme la distance minimale entre u et le point x le plus proche permettant d'avoir $|N_{\text{dist. interne}}(u)| \geqslant$ MinPoints et la distance d'accessibilité (*reachability-distance*) d'un point x_k par rapport à u qui s'interprète comme le maximum entre la distance interne de u et la distance usuelle issue de la métrique employée entre u et x_k.

$$\text{interne}(u) = \begin{cases} \text{Indéfini} \quad \text{Si } |N_\epsilon(u)| < MinPoints \\ d(u, MinPoint^{\text{ème}} \text{ obs.}) \quad \text{Sinon} \end{cases}$$

$$\text{accessibilité}(u, x_k) = \begin{cases} \text{Indéfini} \quad \text{Si } |N_\epsilon(u)| < MinPoints \\ \max(interne(u), d(u, x_k)) \qquad \text{Sinon} \end{cases}$$

L'algorithme OPTICS calcule les valeurs ci-dessus pour chaque observation puis ordonne les points dans l'ordre croissant selon leur distance d'accessibilité par rapport à leur plus proche voisin. On peut alors visualiser un graphique appelé *reachability-plot* où l'axe des absisses représente les données ainsi classées et en ordonnées les distances d'accessibilité associées. On peut ainsi choisir un seuil permettant de définir les différents groupes. Les résultats de classification sont présentés en annexe A.7.

5.3.2.3 *Impacts de l'anonymisation par apprentissage non-supervisé*

Nous avons ainsi testé différents algorithmes de classification non-supervisée que nous avons paramétré de sorte à obtenir des agrégats d'individus homogènes et correspondant également à nos critères d'anonymisation (présentés page 5.3.2. Chaque classe est alors représentée par une unique observation (par exemple le barycentre dans le cas où toutes les variables sont numériques). Le modèle de tarification introduit dans ce chapitre est alors recalibré sur ces nouvelles données qui par construction ne permettent plus d'identifier les individus sous-jacents. Le tableau ci-dessous récapitule les différents résultats.

Algorithme	Paramétrage	Sinistralité tot. Anonymisée	Σ quadratique écarts	Écart (anonymisée – benchmark)	Écart relatif (à benchmark)
K-Means	K=6000	+1.324e7	+1.391e11	+5.779e5	+4.56 %
	K=4500	+1.325e7	+1.392e11	+5.879e5	+4.64 %
	K=3000	+1.388e7	+1.390e11	+1.218e6	+9.62 %
	K=1200	+1.592e7	+1.399e11	+3.258e6	+25.73 %
Hclust	Ward	+1.389e7	+1.394e11	+1.228e6	+9.7 %
OPTICS	(2,2,120)	+1.445e7	+1.391e11	+1.788e6	+14.12 %
	(2,3,80)	+1.407e7	+1.389e11	+1.408e6	+11.12 %
AP	Bloc de 1000	+1.544e7	+1.398e11	+2.778e6	+21.94 %
	Bloc de 500	+1.594e7	+1.401e11	+3.278e6	+25.89 %
	Bloc de 250	+1.521e7	+1.398e11	+2.548e6	+20.12 %
	Bloc de 100	+1.573e7	+1.398e11	+3.068e6	+24.23 %

Ecarts quadratique minimal

Écarts absolu et relatif minimal

Figure 88.: Tableau de synthèse des méthodes d'anonymisation

Tout d'abord, dans l'étude d'impacts, nous avons accordé une importance particulière à la distribution des écarts relatifs à l'issue de l'anonymisation, à la forme du nuage de points comparant le modèle de référence aux modèles anonymisés ainsi qu'à l'écart relatif total observé (en % par rapport au modèle benchmark).

Les différentes méthodes utilisées présentent des résultats intéressants. En effet, chacune d'entre elles s'avère surévaluer le montant de la sinistralité totale et donc en moyenne des primes pures calculées (dans des grandeurs raisonnables $< 5\%$ pour certaines méthodes). Un tel comportement est plutôt rassurant dans un cadre prudentiel puisque l'on obtient avec nos modèles anonymisés des montants de sinistres prévisionnels supérieurs en moyenne aux frais modélisés par le modèle ligne à ligne classique, ce qui pour l'assureur est une erreur davantage acceptable (tant que l'écart reste raisonnable) qu'une erreur risquant de sous-estimer la sinistralité du portefeuille. Cependant, une sur-estimation du prix pourrait être contestable d'un point de vue commercial. Dans le cas où tous les assureurs n'appliqueraient pas la même méthodologie, on s'expose alors à un risque d'anti-selection. Néanmoins, dans le cas où la méthodologie est la même (par exemple imposée par un régulateur), cette étude nous permet d'estimer un premier ordre de grandeur de l'impact à savoir une augmentation de la prime pure[7].

Sur le tableau 88, on constate ainsi que l'utilisation de K-means semble être la moins impactant sur le modèle de tarification. Le prix obtenu sur les données ainsi anonymisées, n'est alors en moyenne que de $+4,6\%$ en écart relatif par rapport à la prime pure calibreé sur les données d'origine. La nature de la méthode de K-means s'appuyant sur une optimisation d'espérance permet sans doute d'expliquer ce résultat peu divergeant sur

7 Cette valeur représente qu'une simple observation. Il faudrait pouvoir effectuer cette mesure sur plusieurs échantillons de données qui malheureusement n'étaient pas en notre possession lors de l'étude.

notre modèle de tarification qui repose également sur un calcul d'espérance.

L'analyse des écarts a été approfondi à la maille individuelle. Cela nous a notamment permis de mieux comprendre l'hypothétique impact de l'anonymisation sur les assurés [Ly and Poinsignon 2019]. La Figure 89 permet par exemple de mieux comprendre les profils des assurés les plus impactés par une telle anonymisation. S'il faut croiser ces analyses avec les différentes expositions des profils dans la base, on peut néanmoins souligner qu'il existe une sur-représentation des catégories de retraités sur les lignes où le modèle d'anonymisation à le plus d'écart.

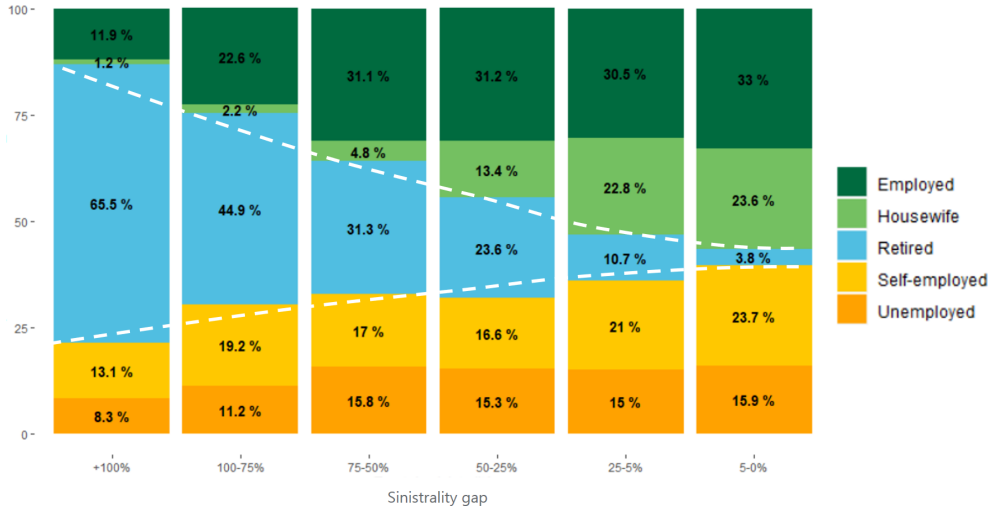


Figure 89.: Analyse des profils d'assurés selon les écarts avec le modèle ligne-à-ligne de référence

Ce constat nous a amené à lancer une nouvelle étude afin de mieux comprendre cet effet. Par ailleurs, le nuage de points en Figure 90 nous renseigne sur le fait qu'une anonymisation par agrégation avec un K-Means par exemple nous amène à sous-estimer les grands risques et sur-estimer les sinistres moyens plus nombreux. Cela s'explique alors assez naturellement par le fait que notre méthode effectue une mutualisation locale des risques lors de la construction de "l'individu" anonyme.

Au-delà des résultats numériques, la comparaison des différentes méthodes nous a également permis d'établir différents critères d'analyse des algorithmes. Nous avons ainsi établi quatre catégories d'impacts que nous schématisons en Figure 91 et qui permettent ainsi d'arbitrer les différentes méthodes au regard de leurs impacts sur le modèle de tarification.

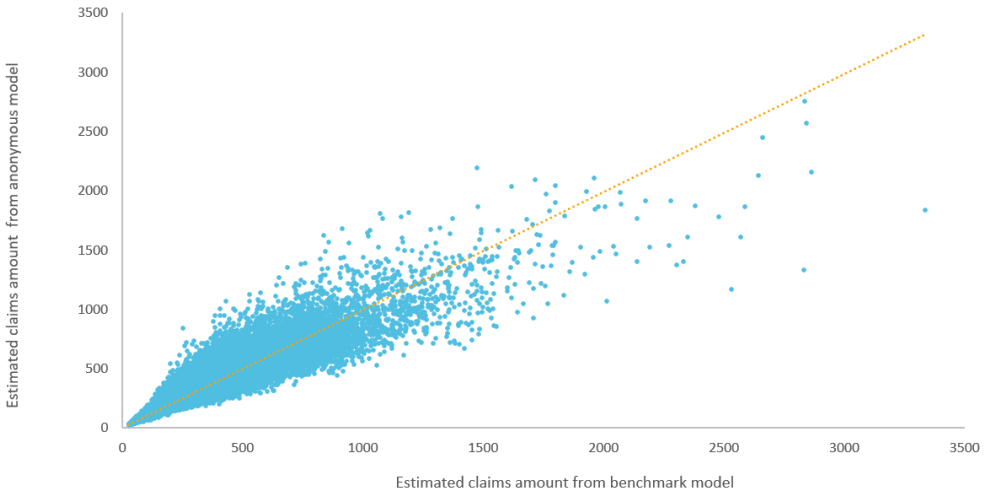


Figure 90.: Prédiction du modèle anonymisé vs modèle de référence

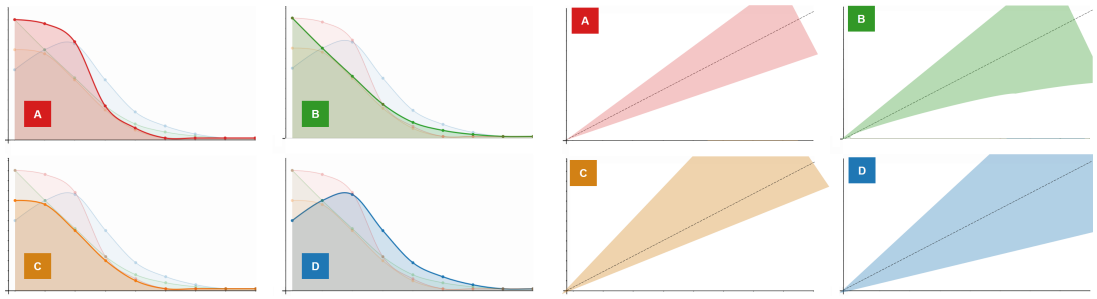


Figure 91.: Schéma des distributions des écarts relatifs & des nuages de points en fonction des catégories des modèles anonymisés

La première d'entre elles (A) correspond aux impacts individuels moyens faibles. On y retrouve par exemple les méthodes OPTICS. La catégorie (B) regroupe les modèles permettant d'obtenir le plus faible impact en moyenne (i.e. assurant une bonne mutualisation). On y retrouve par exemple le modèle K-Means présentés plus haut. La classe (C) représente les méthodes avec un impact positif plus élevé sur la tarification, tant du point de vue individuel que du point de vue global. Cela peut être dommageable pour les assurés ou pour les assureurs si l'effet n'est pas général à tous les acteurs (risque d'anti-selection). On retrouve dans cette classe par exemple la propagation d'affinité (hors celle testée AP_1000). La dernière classe (D) est sans la catégorie non désirable qui présente un impact négatif tant à l'échelle individuelle que globale sur le modèle. C'est l'exemple des résultats obtenus avec AP_1000.

5.3.3 *Limites et approfondissement des travaux*

Il existe plusieurs limitations à la méthode de tarification que nous avons proposé. Tout d'abord les résultats présentés précédemment sont issus d'un type de modèle de tarification basé sur des méthodes linéaires (GLM) afin de modéliser les coûts moyens des sinistres ainsi que leurs fréquences. Cependant d'autres méthodes statistiques peuvent être employées à ces fins, notamment des algorithmes reposant sur la construction d'arbres de décisions tels que le CART ou le random forest, ainsi que le gradient boosting. Nous avons alors essayé d'appliquer notre procédure de tarification anonymisée sur un nouveau modèle de référence dont la sinistralité et la fréquence sont cette fois modélisés à partir de forêts d'arbres de décisions (RF), et également sur un second modèle basé sur des gradient boosting (GB). Les travaux sont toujours en cours afin de pouvoir obtenir une sensibilté à la famille des modèles utilisées comme référence.

Les analyses des modèles présentées dans la partie précédente sont enfin dépendants de notre cadre d'étude et des données utilisées. L'étude doit donc être perçue comme une approche méthodologique et permet de fournir un premier ordre de grandeur dans le cadre spécifique de la tarification automobile sur un portefeuille similaire à celui que nous avons étudié.

De manière générale, les travaux présentés dans ce chapitre nous ont permis d'appréhender la réglementation RGPD et de pouvoir apporter un regard quantitatif quant à l'application rigoureuse de ses termes sur un modèle assurantiel. Au-delà de s'approprier également différents modèles de classification non-supervisée, ces travaux ont permis de mettre en lumière différentes problématiques associées à l'anonymisation des données. La première, qui fait l'objet d'approfondissement est celle de la sensibilité aux différentes familles de modèles à données fixées.

Par ailleurs, l'étude d'impacts amène également le sujet de la capacité à interpréter et mieux comprendre les effets d'une modélisation. Ce devoir d'interprétation des algorithmes est explicité par la réglementation RGPD et nous a amené à nous interroger sur cette notion et les solutions que la littérature apporte aujourd'hui à cette question.

6

L'ENJEU DE LA TRANSPARENCE DES MODÈLES

Au-delà de la protection des données à caractère personnel, le RGPD introduit depuis mai 2018 certaines obligations aux industries à l'égard des utilisateurs de modèles. Il impose en effet d'être en mesure de démontrer à tout moment la maîtrise de ces derniers et ce à tout niveau de leur utilisation. Ainsi, il doit être possible de s'assurer qu'un algorithme complexe n'effectue pas de discrimination et d'expliquer son résultat. L'optimisation et l'élargissement du panel de modèles prédictifs, rendu possible par l'évolution des capacités de calcul, amènent les scientifiques à être vigilants sur l'utilisation des modèles et à considérer de nouveaux outils pour mieux comprendre les décisions de leurs algorithmes. Ces dernières années, la communauté s'est particulièrement intéressée à ce sujet entraînant une intensification des publications. L'utilisation de plus en plus fréquente d'algorithmes complexes (*deep learning*, Xgboost, etc.) présentant des performances séduisantes est sans doute l'une des causes de cet intérêt. Ce chapitre présente un état des lieux des méthodes d'interprétation des modèles, sujet auquel nous avons été sensible tout au long de la thèse. Il constitue ainsi une synthèse des travaux menés dans le cadre du groupe de travail de l'Institut des Actuaires[1] et des recherches menées à Milliman. Un article est prévu pour janvier 2020 avec Dimitri Delcaillau et Franck Vermet sur cette thématique.

1 Groupe supervisé par Marc Juilliard

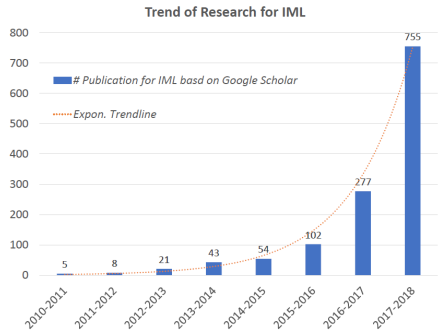


Figure 92.: Nombre d'articles publiés en lien avec l'interprétabilité des modèles de machine learning au cours des 15 dernières années [Adadi and Berrada 2018]

6.1 L'INTERPRÉTABILITÉ DES MODÈLES : UN ENJEU MAJEUR

6.1.1 *Définition de l'interprétabilité*

Si l'on se réfère aux différentes publications de ces trois dernières années, l'interprétabilité des modèles est un nouvel enjeu dans l'utilisation des modèles et plus particulièrement ceux du *machine learning*. Adadi and Berrada (2018) ont en effet montré l'intérêt croissant de la communauté scientifique et des régulateurs pour l'interprétation des modèles. Cependant, bien que le concept d'interprétabilité semble de plus en plus répandu, on note l'absence d'un consensus général tant sur la définition que de la mesure de l'interprétabilité d'un modèle [Molnar 2019]. Il existe effectivement de nombreuses méthodes (graphiques, mathématiques, etc.) qui peuvent être associées à l'interprétation des algorithmes ce qui entraîne parfois une certaine confusion sur cette notion. Par ailleurs, selon l'interlocuteur, il peut contenir différents degrés de définition : parlons-nous de la compréhension du modèle, de la capacité à contrôler les résultats de ce dernier, de sa transparence vis-à-vis d'utilisateurs novices ? Ou faisons-nous référence aux moyens mis en place pour analyser les résultats d'un algorithme aussi complexe soit-il ?

Murdoch and Singh (2019) tentent de donner une définition précise à l'interprétabilité dans le cadre d'un modèle de machine learning. Ils fournissent notamment un cadre (appelé PDR) construit sur trois propriétés souhaitées pour l'évaluation et la construction d'une interprétation : la précision prédictive (P), la précision descriptive (D) et la pertinence (R). Ceci permet de classer les différentes méthodes existantes et d'utiliser un vocabulaire commun entre les différents acteurs du domaine de l'apprentissage statistique.

6.1.2 *Définir un modèle interprétable*

L'article [Murdoch and Singh 2019] suggère tout d'abord que l'interprétation fasse référence à la notion d'extraction d'information. Miller (2017) propose plus précisément de définir l'interprétabilité comme le degré à partir duquel un humain peu comprendre la cause de la décision. Une définition alternative est également proposée par Molnar (2019) : l'interprétabilité est alors "le degré à partir duquel un humain peut régulièrement prédire le résultat du modèle". Ainsi, une connaissance est dite pertinente [Murdoch and Singh 2019] si elle fournit une information pour un public particulier et un problème d'un domaine choisi. La notion d'interprétabilité peut donc s'évaluer selon plusieurs critères :

- **La confiance** : ce critère revient régulièrement. Abordée par exemple dans [Ribeiro et al. 2016], la notion d'interprétabilité mène nécessairement à la capacité d'un modèle à fournir des résultats qu'un décisionnaire peut utiliser. Un modèle doit pouvoir s'utiliser en toute sérénité.
- **La causalité** : bien que l'un des objectifs des algorithmes d'apprentissage statistique soit de mettre en avant des corrélations, l'algorithme doit permettre de mieux comprendre des phénomènes du monde réel et les interactions entre différents facteurs observés.
- **"La transférabilité"** : définie comme la capacité d'un modèle à s'adapter à des situations légèrement différentes, elle est une des propriétés souhaitées dans la recherche d'interprétabilité. Elle transcrit la capacité de généralisation.
- **"L'informativité"** : l'utilisation d'un algorithme doit pouvoir dépasser la simple optimisation mathématique. Un modèle doit pouvoir fournir une information précise sur la prise de décision.
- **Une prise de décision juste et éthique** : ce critère rejoint les directives du RGPD. L'utilisateur d'un algorithme doit pouvoir s'assurer de l'absence de biais dans la prise de décision et un respect de l'éthique (ne pas commettre de discrimination par exemple).

Même s'il est difficile de prendre en compte de manière objective l'ensemble de ces axes, l'interprétabilité peut être garantie par l'utilisation de deux grandes familles d'outils : ceux qui s'appuient sur le modèle lui-même et ceux qui s'appliquent *a posteriori* via des analyses *post-hoc*. Pour bien les choisir, le cadre PDR suggère [Murdoch and Singh 2019] d'utiliser des méthodes qui permettent d'interpréter un modèle sous trois angles : la "précision Prédictive", la "précision Descriptive" et la "Pertinence".

La précision "Prédictive" nous permet d'évaluer si la méthode d'interprétation nous permet de comprendre la performance du modèle. Par exemple, elle nous aide à étudier

la distribution des prédictions : il peut être problématique que l'erreur de prédiction soit bien plus élevée pour une classe spécifique ; ou à s'assurer que le modèle est robuste (i.e. non sensible à l'échantillonnage) et ainsi faire confiance aux relations mises en avant par le modèle.

La précision "Descriptive" est définie dans [Murdoch and Singh 2019] comme le "degré à partir duquel une méthode d'interprétation capture objectivement les relations apprises par les modèles de *machine learning*". En général, les modèles perçus comme interprétables comme les arbres de classification par exemple, ou les régressions linéaires, possèdent des méthodes d'interprétation s'appuyant sur les modèles (graphiques, effet-marginaux, odd-ratio, etc.) avec une bonne précision descriptive. Lors du choix du modèle de *machine learning* pour répondre à une problématique, un compromis est donc à réaliser entre précision prédictive et précision descriptive. Intrinsèquement, les modèles "simples" vont posséder des méthodes d'interprétation à fort pouvoir descriptif alors que les modèles complexes (à forte paramétrisation comme le *deep learning* ou à grande profondeur comme le Xgboost) posséderons une bonne précision prédictive et nécessite alors des interprétations *post-hoc* afin d'augmenter la précision descriptive.

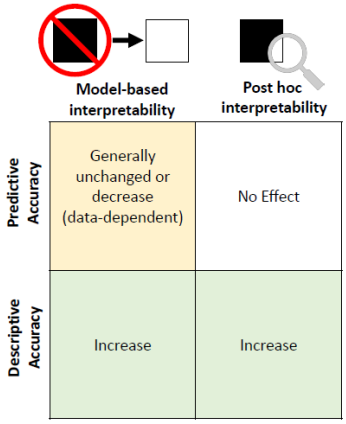


Figure 93.: Impact des méthodes d'interprétabilité sur les précisions prédictive et descriptive dans le cadre du PDR [Murdoch and Singh 2019]

Enfin, la "Pertinence" de l'information apportée par l'interprétation du modèle est également cruciale. Ainsi, "une interprétation est dite pertinente si elle fournit des informations pour un public particulier et un domaine choisi" [Murdoch and Singh 2019]. L'interprétabilité est donc également dépendante du public concerné par le modèle : un décisionnaire, un médecin, un patient, un assuré, etc. Ce dernier critère permet alors parfois d'arbitrer entre la précision "Prédictive" et "Descriptive". L'interprétabilité se mesure donc par une approche méthodique et le choix d'outils adaptés afin d'assurer la

bonne compréhension des résultats obtenus via un processus de modélisation. Comme introduit précedemment, l'interprétabilité peut donc s'étudier dans un premier temps sous deux niveaux : l'interprétabilité basée sur les modèles et l'interprétabilité *post-hoc* (agnostique aux modèles).

6.1.3 *Les deux niveaux d'interprétabilité*

6.1.3.1 *L'interprétabilité basée sur le modèle (IBM)*

L'interprétabilité basée sur le modèle (IBM), constitue le premier niveau d'interprétabilité. Il intervient pendant l'élaboration du modèle et est lié au choix des familles d'algorithmes utilisées pour comprendre un phénomène et leur calibrage. Un modèle interprétable peut alors se définir par sa :

- parcimonie : La parcimonie est étroitement liée au principe du rasoir d'Ockham, qui stipule que "les multiples ne doivent pas être utilisés sans nécessité". Dans le cas d'un modèle de *machine learning*, imposer que le modèle soit parcimonieux revient à limiter le nombre de paramètres non nuls. En statistique comme en apprentissage automatique il existe alors différentes méthodes de régularisation comme celles introduites au chapitre 3 ou en annexe A.2, applicables à de nombreux modèles[2].

- simulabilité : Murdoch and Singh (2019) définit un modèle comme simulable si un humain est capable de reproduire le processus de décision global de l'algorithme. Ainsi la simulabilité réfère à une transparence totale du modèle : un humain devrait être capable, à partir des entrées et des paramètres du modèle, de réaliser l'ensemble des calculs, en temps raisonnable, pour reconstruire la prédiction faite par le modèle. En ce sens, les arbres de décision sont généralement cités comme des algorithmes simulables, étant donné leur simplicité visuelle pour la prise de décision. De même, les règles de décision appartiennent à cette catégorie.

- modularité : un modèle est modulaire si une portion significative du processus de prédiction peut être interprétée indépendamment. Ainsi, un modèle modulaire ne sera pas aussi simple à comprendre qu'un modèle parcimonieux ou simulable mais peut augmenter la précision descriptive en fournissant des relations apprises par l'algorithme. Un exemple classique de modèle considéré comme modulaire est la famille des GAM (modèles additifs généralisés) [Tibshirani 1990], dont les GLM sont une sous-famille. Dans ce type de modèles, la relation entre les variables est forcément additive et les coefficients trouvés permettent une interprétation relativement facile du modèle. Par opposition, les réseaux de neurones profonds sont eux considérés comme peu modulaires, étant donné le peu d'informations

2 Le Xgboost introduit une méthode de régularisation tout comme le *deeplearning* avec le *drop-out*. Cependant même sous contrainte, ces modèles sont souvent peu parcimonieux.

fournies par les coefficients de chaque couche. L'exemple cité précédemment sur la prédiction des décès à cause de la pneumonie montre l'intérêt de la modularité pour produire des interprétations pertinentes, pour détecter des biais dans la base d'apprentissage.

Selon sa nature, un modèle possède alors des propriétés et des outils d'analyse qui vont permettre une compréhension plus ou moins simple selon les points mentionnés précédemment. Le second niveau d'interprétation est moins sensible aux algorithmes utilisés lors du processus de modélisation.

6.1.3.2 *L'interprétabilité post-hoc*

L'interprétabilité *post-hoc*, à la différence de l'interprétabilité basée sur le modèle, correspond à l'analyse une fois que le modèle ait été ajusté. Cette interprétation *a posteriori* intervient dans le but de fournir des informations sur les relations éventuellement capturées par l'algorithme. Ce type d'interprétations est sans doute celui où la recherche s'est le plus activée ces dernières années. Il s'avère particulièrement utile pour analyser des modèles complexes mais à forte précision prédictive.

L'analyse *post-hoc* vient alors augmenter la précision descriptive du modèle. Elle intervient plus particulièrement à deux niveaux : sur la compréhension du modèle au regard des données utilisées et sur l'analyse des prédictions fournies par l'algorithme. Elle est donc un supplément aux modèles utilisés. Ces méthodes ont connu ces dernières années des évolutions assez prononcées permettant de dépasser les limites des outils pré-existants d'analyse notamment celles des arbres [Breiman 2001b] ou des réseaux de neurones [Olden et al. 2004].

L'interprétation au niveau des données permet de s'intéresser aux relations générales apprises par le modèle, c'est-à-dire aux règles pertinentes d'une classe particulière de réponses ou d'une sous-population. De ces deux niveaux d'interprétation *post-hoc* se dégagent alors des outils d'interprétation globaux et locaux. La section qui suit présente différentes méthodes d'interprétation *post-hoc*.

6.2 LES MÉTHODES D'INTERPRÉTATION *post-hoc*

Dans cette section nous nous intéressons aux méthodes *post-hoc* agnostiques aux modèles. Il existe bien évidemment des méthodes propres à chaque algorithme renforçant l'interprétation de ces derniers (comme par exemple l'importance des variables des arbres [Breiman 2001b]) mais nous nous y intéressons pas dans ce chapitre. La Figure 94 résume la répartition des méthodes d'interprétations selon différents niveaux d'un processus de modélisation. Nous détaillons dans les sections suivantes les outils d'interprétation qui

nous ont semblé les plus pertinents au regard des modèles complexes les plus couramment utilisés en assurance à savoir les modèles par arbres (Random Forest ou Gradient Boosting). Ces méthodes sont par ailleurs applicables à tout algorithme.

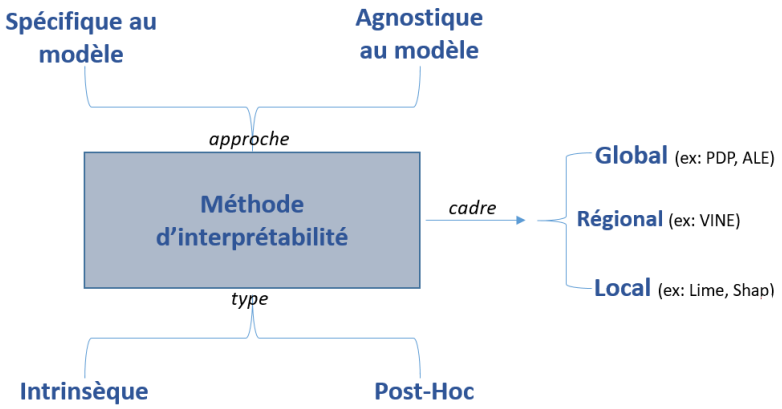


Figure 94.: Différentes catégories d'interprétabilité des modèles

6.2.1 *Méthodes graphiques d'interprétation*

6.2.1.1 *Graphique de dépendance partielle (PDP)*

L'analyse PDP (Partial Dependance Plot), introduite par [Friedman 2001] est sans doute la méthode la plus ancienne d'inteprétation des modèles au regard des publications de ces trois dernières années. Cette méthode graphique de dépendance partielle a pour objectif de montrer l'effet marginal d'une ou plusieurs variables explicatives sur la prédiction faite par un modèle. C'est une méthode d'interprétation globale.

Considérons un modèle $\hat{f}$ entraîné sur une base d'apprentissage $Z^{(i)} = (X^{(i)}, Y^{(i)}) \in \mathbb{R}^p \text{x} \mathbb{R}$, associées aux observations $(x^{(i)}, y^{(i)})$ avec $x^{(i)} = (x_j^{(i)})_{1 \leq j \leq p}$ pour $1 \leq i \leq n$ (où $p, n \in \mathbb{N}$). Notons X_S l'ensemble des variables pour lesquelles on veut connaître l'effet sur la prédiction et X_C les variables explicatives restantes. Par exemple $X_S = (X_1, X_2)$ et $X_C = (X_3, ..., X_p)$. Ainsi, $X = (X_S, X_C)$, représente l'ensemble des variables explicatives utilisées par notre modèle.

On définit alors la fonction de dépendance partielle par la formule:

$$\hat{f}_{x_S}(x_S) = \mathbb{E}_{X_c}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_s, x_c) d\mathbb{P}_{X_c}(x_c) \tag{30}$$

Notons que cette formule diffère de l'espérance conditionnelle de X_S. Afin de l'estimer, on utilise la méthode de Monte Carlo sur la base des n observations :

$$\hat{f}_{x_S}(x_S) \simeq \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}) \tag{31}$$

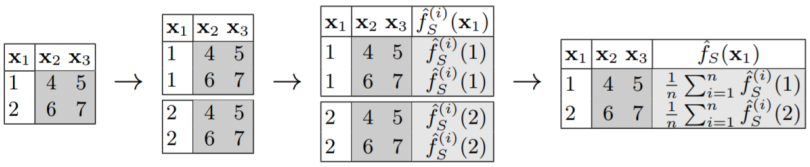


Figure 95.: Calcul du graphique PDP sur un exemple simple

L'algorithme de construction de la courbe est détaillé en annexe A.10. Il repose sur l'hypothèse forte de non corrélation entre les variables de l'ensemble C et celles de S. Dans la pratique, ce cas est rarement vérifié ce qui mène à la considération d'associations de modalités non possibles en réalité (par exemple d'observer un individu de 2m avec un poids inférieur à 10kg si on considère des variables de morphologies).

Afin d'illustrer la méthode, considérons l'exemple qui introduit les variables ci-après.

$$Y = 0.2X_1 - 5X_2 + 10X_2\mathbb{1}_{\{X_3 \geq 0\}} + \epsilon$$

Où $X_1, X_2, X_3 \overset{iid}{\sim} U(-1,1), \epsilon \sim N(0,1)$ avec ϵ indépendant de X_1, X_2, X_3. On suppose qu'on observe un échantillon de taille $n = 1000$. Le nuage de points (*scatter-plot*) associé à X_2 et Y de cet échantillon est représenté à gauche de la Figure 96. Le graphique de PDP de la variable X_2 associé au modèle de *Random Forest* mis en place afin de prédire Y est représenté à droite de la Figure 96.

La Figure 96 de PDP suggére qu'en moyenne la variable X_2 n'est pas significative dans la prédiction de Y par le modèle de forêt aléatoire alors que le nuage de points semble suggérer une conclusion inverse. Ce cas illustre la problématique de la non prise en compte des corrélations entre les variables par PDP. Il existe alors une alternative à cette méthode PDP appelée ICE, détaillée en annexe A.11.

En plus de donner l'effet marginal moyen d'une variable, les graphiques de PDP peuvent fournir une information sur l'importance d'une variable dans la prédiction faite par un modèle. En effet, lorsque le graphe de PDP associé à la variable X_1 (par exemple) est relativement plat, il semble naturel de penser que cette variable n'a pas beaucoup d'influence sur la prédiction de Y. L'idée introduite par Greenwell (2018) est ainsi de

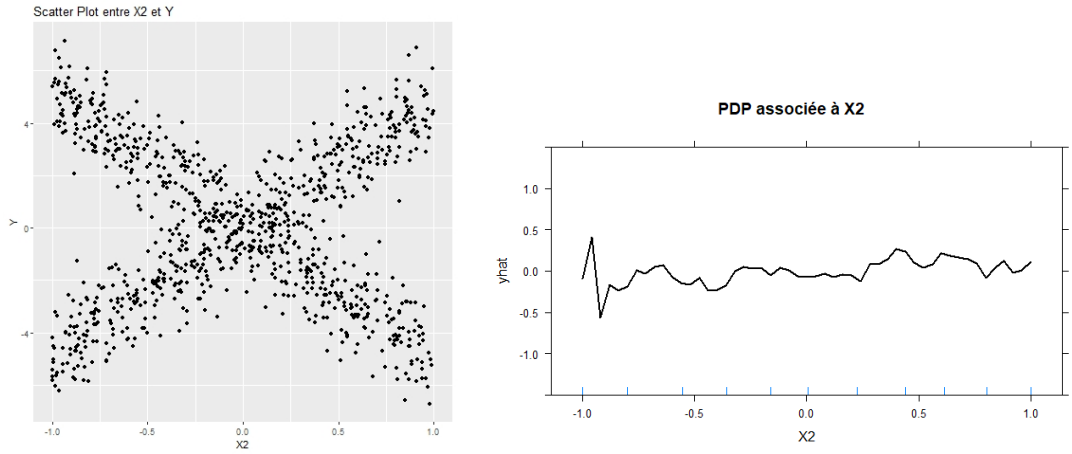


Figure 96.: Scatter Plot de X_2 et Y (à gauche) et graphique PDP de X_2 (à droite)

définir une fonction $Flat$ qui mesure la "platitude" de la courbe de PDP : pour une observation x, $i(x) = Flat(\hat{f}_{x_s}(x_s))$. Une mesure simple et efficace proposée [Greenwell 2018] est la variance empirique lorsque les variables x_S sont continues et la statistique d'intervalle divisée par 4 pour les variables catégorielles à $K \in \mathbb{N}$ niveaux. Dans le cas où $S = 1$, on obtient alors les formules :

$$i(x_1) = \begin{cases} \frac{1}{n-1} \sum_{i=1}^{n} [\hat{f}_{x_1}(x_1^{(i)}) - \frac{1}{n} \sum_{i=1}^{n} \hat{f}_{x_1}(x_1^{(i)})]^2 \ si\ x_1\ est\ continue \\ [\underset{1 \leq i \leq n}{max} \hat{f}_{x_1}(x_1^{(i)}) - \underset{1 \leq i \leq n}{min} \hat{f}_{x_1}(x_1^{(i)})]/4 \ si\ x_1\ est\ qualitative \end{cases}$$

Cette technique est appelée IPD (Importance Based On Partial Dependance). En outre, le graphique PDD permet de fournir une meilleure interprétation des relations entre la variable à expliquer par l'algorithme et les variables endogènes de la base de données.

Ainsi, le graphique PDP est souvent utilisé pour sa simplicité d'interprétation et sa facilité d'implémentation. De plus, ce graphique peut servir d'outil dans l'estimation de l'importance des variables et leurs interactions au sein d'un modèle. Cependant, ce graphique seul ne suffit pas à expliquer toute la complexité d'un algorithme. La méthode de calcul repose effectivement sur une hypothèse forte et limitante d'indépendance entre les variables. Par ailleurs, le PDP masque les effets hétérogènes comme illustré sur la Figure 96. C'est la raison pour laquelle cette méthode est souvent associée à d'autres graphiques comme ICE détaillé en annexe A.11.

6.2.1.2 *Graphique des effets locaux accumulés (ALE)*

Le graphique des effets locaux accumulés (Accumulated Local Effects Plot) a pour objectif de corriger les limites de PDP, notamment lorsque les variables explicatives sont corrélées. La méthode a été introduite par Apley (2016). Tout comme PDP, l'ALE est une approche globale d'interprétation.

Partant de l'exemple des combinaisons de poids et tailles introduit par Molnar (2019), le PDP illustré en Figure 97 ne tient effectivement pas compte de la distribution empirique.

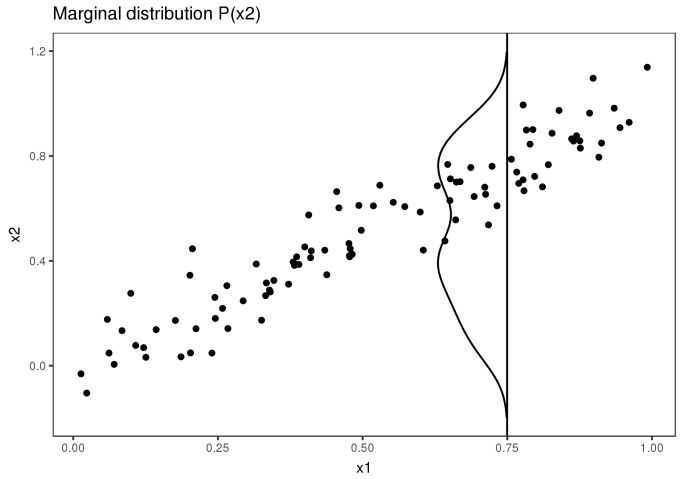


Figure 97.: Cas du calcul de la PDP avec des variables très corrélées lorsque l'on fixe x_1=0.75 [Molnar 2019]

Une première idée illustrée en Figure 98 afin d'éviter ce problème est alors, dans le calcul de PDP, de moyenner à partir de la distribution conditionnelle, ce qui signifie que pour une valeur x_1 donnée, on réalise la moyenne des instances avec des valeurs similaires localement à x_1.

Cependant cette approche ne permet pas de tenir compte d'effets combinés de variables. Par exemple, si on souhaite prédire Y le prix d'une maison, à partir des variables X_1 (surface de la maison) et X_2 (nombre de chambres). En supposant que la variable de surface de la maison n'a pas d'effet sur la prédiction mais que seul le nombre de chambres en a un, comme le nombre de chambres augmente avec la surface, le M-plot précédent montrera alors que la surface de la maison fait augmenter son prix.

Le graphique des effets locaux accumulés (ALE) dépasse cette limite. Ce dernier repose sur la distribution conditionnelle des variables mais calcule également les différences en prédiction à la place de moyennes. Ainsi si l'on veut comprendre l'effet associé à une surface de 30m^2, la méthode ALE utilise toutes les instances (maisons) de 30m^2 et

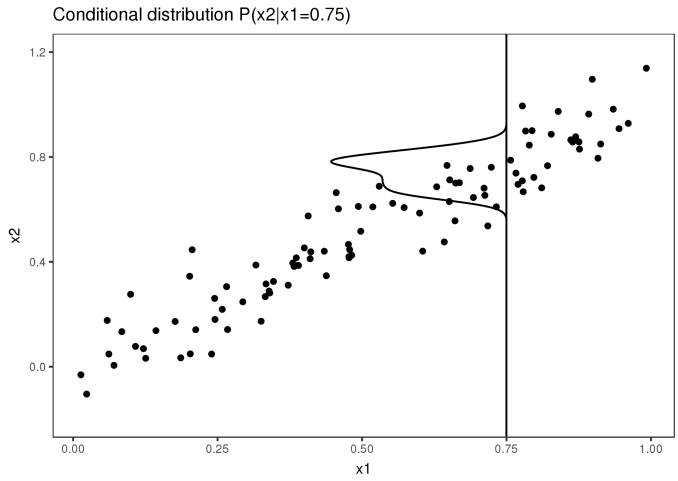


Figure 98.: M-plot dans le cas de deux variables très corrélées en utilisant la distribution conditionnelle de x_2 sachant $x_1 = 0.75$ (issu du site [**??**])

regarde la différence en prédiction lorsque l'on change sa surface de 31m^2 à 29m^2. Ceci donne alors l'effet de la variable de surface et non l'effet combiné avec les variables corrélées comme dans le M-plot. Le graphique 99 résume l'idée de calcul de l'ALE : on divise tout d'abord la variable X_1 en intervalles ; pour chaque instance dans un intervalle, on calcule la différence en prédiction lorsqu'on remplace la valeur de X_1 par la borne supérieure et inférieure de l'intervalle considéré ; enfin, toutes ces différences sont accumulées et centrées donnant alors la courbe d'ALE.

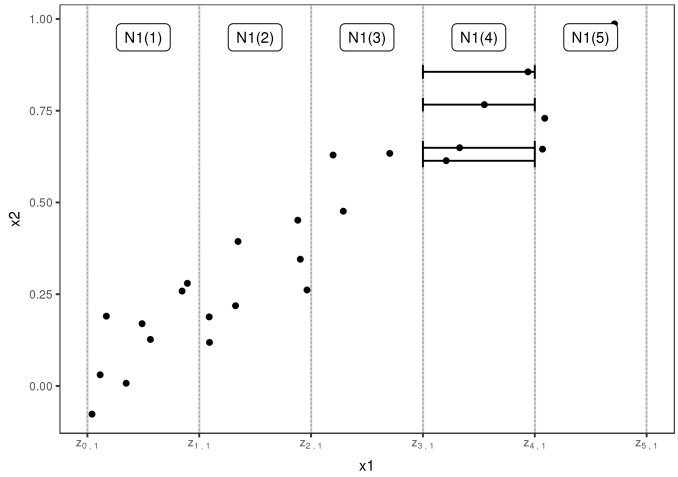


Figure 99.: Explication du calcul de l'ALE avec des variables X_1 et X_2 très corrélées [Molnar 2019]

La méthode de construction de la courbe est précisée en annexe A.12. Nous venons ainsi d'illustrer deux méthodes qui permettent d'analyser de manière globale tout modèle. Cependant, lorsque les modèles sont complexes ou pas suffisamment parcimonieux, ces outils ne sont parfois pas suffisant pour comprendre les prédictions. Les deux parties suivantes introduisent alors deux algorithmes récemment publiés proposant des méthodes d'interprétation locales : LIME [Ribeiro et al. 2016] et SHAP [Lundberg and Lee 2017].

6.2.2 *LIME*

LIME [Ribeiro et al. 2016] est l'une des premières approches locales apparues dans le domaine du machine learning interprétable. Cette méthode consiste à utiliser un modèle de substitution (noté M_2) qui approche localement le modèle que l'on tente d'expliquer (noté M_1).

Cette substitution s'effectue en appliquant dans un premier temps une légère perturbation des données initiales X. On crée alors un nouvel échantillon, noté $\tilde{X}$. Sur ce dernier, on applique alors le modèle M_1 afin de reconstruire la variable à expliquer correspondante. Ainsi, on notera M_1: $\hat{y_1} = f_1(\hat{X})$. Chaque observation de l'échantillon $\tilde{X}$ simulé est ensuite pondéré en fonction de sa proximité avec les données initiales : plus celui-ci est proche, plus son poids est important. Sur ces données pondérées, on construit alors un modèle simple d'interprétation M_2. Celui-ci est généralement de type Lasso pour la régression et un arbre de décision pour la classification. Notons que cette fois-ci le modèle M_2 fournit une bonne approximation locale mais pas nécessairement une approximation globale.

Ainsi la fonction $\hat{g}$ associée au modèle M_2 est trouvée en résolvant le problème d'optimisation :

$$\hat{g} = \underset{g \in G}{argmin}[J(f, g, \pi_x) + \Omega(g)] \tag{32}$$

avec J la fonction de coût ; f la fonction associée au modèle M_1, g la fonction associée au modèle M_2 qu'on souhaite optimiser, appartenant à la classe de modèle G ; π_x une mesure de proximité définissant la taille du voisinage autour de x que nous considérons pour l'interprétation du modèle et Ω une fonction traduisant la complexité d'un modèle.

En pratique, LIME n'optimise que le terme associé à la fonction de coût. Il convient à l'utilisateur de choisir un modèle peu complexe, comme par exemple si f_1 est une régression, un modèle avec un nombre limité de variables explicatives (Cf critère de parcimonie).

La Figure 100 [Ribeiro et al. 2016], résume le fonctionnement de LIME dans le cas d'un modèle de classification binaire (classe 0 ou 1), avec deux variables explicatives. La

zone en bleu représente les points associés à la classe 1 selon le modèle étudié et la zone rose clair les points associés à la classe 0. Les croix roses et les points bleus représentent quant à eux les données simulés pour l'apprentissage du modèle de substitution. La taille du motif représente le poids du point considéré, suivant sa distance à l'observation d'intérêt, représenté par la croix rouge. La droite grise en pointillés est la limite de décision obtenue par l'algorithme LIME à l'aide d'un modèle linéaire.

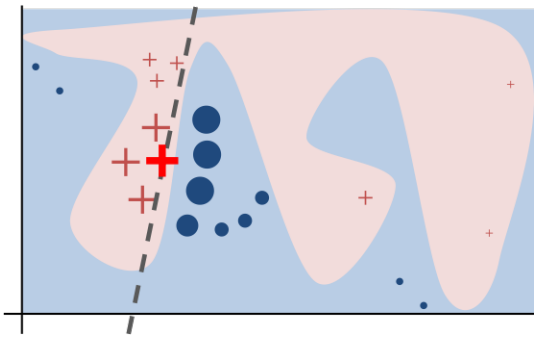


Figure 100.: Principe de LIME

Comme le remarque T. Laugel (2018), le choix du noyau dans l'algorithme LIME est primordial. Il peut en effet avoir un impact majeur sur la fidélité et la précision de l'explication qui en découle. Par exemple, si on considère une variable explicative X et un modèle de décision représenté par le trait noir sur la Figure 101. Notre objectif est de comprendre localement la prédiction faite par le modèle au niveau de l'instance $x = 1.6$ (représentée par la croix noire). Les lignes tracées de différentes couleurs (jaune, vert et violet) illustrent la sensibilité de l'approximation locale de LIME au paramètre σ du noyau. On observe effectivement que les lignes jaunes (σ=2) et vertes (σ=0.75) répliquent peu le comportement local du modèle contrairement à la courbe violette (σ=0.1). L'annexe A.13 détaille alors une extension de LIME proposée par T. Laugel (2018).

Tan et al. (2019) soulignent les incertitudes liées aux méthodes d'interprétation locales des modèles de machine learning, dont LIME. Les auteurs mettent en garde les utilisateurs sur la robustesse et la confiance que l'on peut en avoir en la méthode. L'utilisation de LIME semble alors déplacer la question de l'interprétabilité des modèles complexes sur les outils eux-mêmes utilisés pour la résoudre. Tan et al. (2019) soulignent notamment via un exemple :

- *Le hasard dû à l'échantillonnage des données* : il s'agit de la variance d'échantillonnage pour expliquer un unique point de la base.
- *La sensibilité selon le choix des paramètres* : taille de l'échantillon et la proximité d'échantillonnage.
- *La variation de la crédibilité d'interprétation selon les points étudiés.*

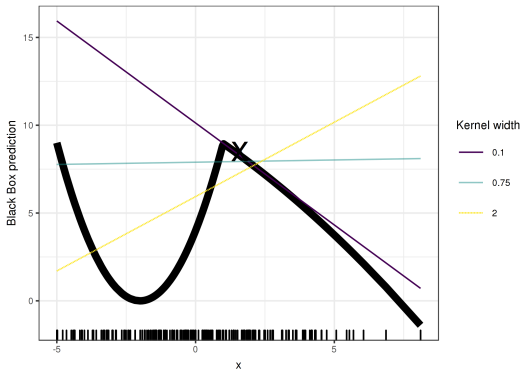


Figure 101.: Choix du paramètre du noyau pour la mesure de proximité dans l'algorithme LIME

Même si LIME présente certaines limites, la méthode permet néanmoins de proposer un premier outil d'interprétation local et permet notamment de compléter les limites des méthodes standards telles que l'importance des variables dans les méthodes par arbres (Random Forest, Gradient Boosting, etc.)

6.2.3 *SHAP*

SHAP [Lundberg and Lee 2017] est également un algorithme d'interprétation locale. Il s'appuie sur la mesure de Shapley introduite en théorie des jeux.

6.2.3.1 *Valeur de Shapley en théorie des jeux*

Quand un modèle réalise une prédiction, intuitivement nous percevons que chaque variable ne joue pas le même rôle. Certaines n'ont même quasiment aucun impact sur la décision prise par le modèle. L'objectif de l'algorithme SHAP est, en s'appuyant sur la valeur de Shapley, de quantifier le rôle de chaque variable dans la décision finale du modèle. Introduisons dans un premier temps la valeur de Shapley [Winter 2002].

Considérons un jeu J caractérisé par un 2-uplet : $J = (P, v)$ où $P = (\{1, ..., p\}$ est un ensemble de p joueurs, $p \in \mathbb{N}^*$), et $v : S(P) \to \mathbb{R}$ est une fonction caractéristique telle que : $v(\emptyset) = 0$ avec $S(P)$ l'ensemble des sous-ensembles de P.

Un sous-ensemble de joueurs $S \in S(P)$ est appelé coalition et l'ensemble $\{1, ..., p\}$ de tous les joueurs est appelé la grande coalition. La fonction caractéristique v décrit l'importance de chaque coalition.

L'objectif du jeu est alors de répartir l'importance de chaque joueur dans le gain total de la manière la plus "juste" possible. Ainsi, on cherche un opérateur ϕ, qui assigne au jeu $J = (\{1, ..., p\}, v)$, un vecteur $\phi = (\phi_1, ..., \phi_p)$ de gains. Comment définir la notion de répartition juste entre les joueurs ? Elle a été introduite par Lloyd Shapley en 1953, à travers quatre axiomes :

- Efficacité : $\sum_{i=1}^{p} \phi_i(v) = v(\{1, ..., p\})$
- Symétrie : Pour tout couple de joueurs $(i, j) \in \{1, ..., p\}^2$, si $\forall S \in S(\{1, ..., p\} \backslash \{i, j\})$, $v(S \cup i) = v(S \cup j)$, alors $\phi_i(v) = \phi_j(v)$
- Facticité : Soit $i \in \{1, ..., p\}$ un joueur. Si $\forall S \in S(\{1, ..., p\} \backslash \{i\}), v(S \cup \{i\})$, alors: $\phi_i(v) = 0$.
- Additivité : Pour tout jeu, $v : S(P) \to \mathbb{R}$, $w : S(P) \to \mathbb{R}$, $\phi(v + w) = \phi(v) + \phi(w)$ avec : $\forall S \in S(\{1, ...p\}), (v + w)(S) = v(S) + w(S)$

La valeur de Shapley ϕ est alors l'unique valeur "juste" qui distribue le gain total $v(\{1, ..., p\})$, c'est-à-dire celle qui respecte les quatre conditions précédentes. Il s'agit d'un théorème démontré par Shapley, pour lequel nous avons également une formule explicite pour cette valeur de Shapley, à savoir:

$$\forall i \in \{1, ..., p\},\ \phi_i(v) = \sum_{S \in S(\{1,...,p\}) \backslash \{i\}} \frac{(p - |S| - 1)!|S|!}{p!} (v(S \cup \{i\}) - v(S)) \tag{33}$$

On peut également définir cette valeur de Shapley d'une autre manière :

$$\forall i \in \{1, ..., p\},\ \phi_i(v) = \frac{1}{p!} \sum_{O \in Perm(P)} v(Pre^i(O) \cup \{i\}) - v(Pre^i(O)) \tag{34}$$

avec : $Perm(P)$ l'ensemble des permutations de $P = \{1, ..., p\}$ et $Pre^i(O)$ l'ensemble des joueurs qui sont prédécesseurs du joueur i dans la permutation $O \in Perm(P)$ (il s'agit du nombre qui apparaît avant le nombre i dans la permutation O).

6.2.3.2 *Valeur de Shapley appliquée à l'interprétabilité des modèles*

SHAP [Lundberg and Lee 2017] utilise ainsi la valeur de Shapley afin d'expliquer le poids de chaque variable dans les prédictions faites par un modèle et ce quelle que soit sa complexité.

On considére ainsi une variable numérique à prédire $Y \in \mathbb{R}$, à partir d'un vecteur $X \in \mathbb{R}^p$ de $p \in \mathbb{N}$ variables explicatives.

On suppose que l'on dispose d'un échantillon: $y = (y_1, ..., y_n) \in \mathbb{R}^n$ correspondant aux valeurs cibles et $x = (x_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$ correspondant aux variables explicatives (avec $n \in \mathbb{N}$ le nombre d'individus).

Notre algorithme M de machine learning est calibré sur cet échantillon et on note $\hat{g}$ la fonction associée au modèle, c'est-à-dire la fonction qui renvoie la prédiction $\hat{y}$ de y faite par le modèle à partir du vecteur x: $\hat{y} = g(x)$.

Si on fait l'analogie avec la théorie de jeu associée à la mesure de Shapley, nous obtenons :

- le jeu : la tâche de prédiction pour une instance $\tilde{x} \in \mathbb{R}^p$ du dataset.
- le gain : la prédiction actuelle de cette instance moins la prédiction moyenne de toutes les instances du jeu de données
- les joueurs : les valeurs des caractéristiques $x_j,\ j \in \{1, ..., p\}$, qui collaborent pour recevoir le gain (ici il s'agit de prédire une certaine valeur)

Supposons que notre variable Y à expliquer soit le prix d'une voiture en euros. Nos variables explicatives sont x_1 et x_2, respectivement le nombre de chevaux de la voiture et le nombre de portes. Supposons que pour $x_1 = 150$ et $x_2 = 4$, nous ayons un prix estimé par le modèle $\hat{g}$ de : $y = 150000$. Nous savons également qu'à partir des données initiales (constitué de plusieurs prix de voitures et des variables explicatives associées), nous avons une prédiction moyenne de 170000 euros.

L'objectif du jeu est alors d'expliquer cette différence de -20000 euros, entre la prédiction faite par le modèle et la prédiction moyenne. On pourrait par exemple, obtenir le résultat suivant: x_1 a contribué pour $+10000$ euros et x_2 pour -30000 euros (par rapport à la valeur moyenne prédite) et justifierait donc la différence de -20000 euros observée.

Finalement, on peut définir la valeur de Shapley comme la contribution marginale moyenne d'une variable (explicative) sur toutes les coalitions possibles.

6.2.3.3 *Valeur de Shapley dans le cas de la régression linéaire*

On considère le modèle linéaire : $\hat{g}(x) = \beta_0 + \sum_{i=1}^{p} \beta_i\, x_i$, avec $(\beta_i)_{0 \leq i \leq p} \in \mathbb{R}^p$. On définit alors la valeur de Shapley de la variable $j \in 1, ..., p$ associée à la prédiction $\hat{g}(x)$: $\phi_j(\hat{g}) = \beta_j x_j - \mathbb{E}[\beta_j X j] = \beta_j(x_j - \mathbb{E}[Xj])$ (avec $\mathbb{E}[\beta_j X j]$ l'effet moyen de la variable x_j). On parle aussi de contribution de la variable x_j dans la prédiction de $\hat{g}(x)$, car il s'agit de la différence entre l'effet de la variable et l'effet moyen. On peut remarquer

que la somme des contributions de toutes les variables explicatives donnent la différence entre la valeur prédite pour x et la valeur de prédiction moyenne. En effet :

$$\sum_{j=1}^{p} \phi_j(\hat{f}) = \sum_{j=1}^{p} (\beta_j x_j - \mathbb{E}[\beta_j X_j]) = (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} \mathbb{E}[\beta_j X_j]) = \hat{g}(x) - \mathbb{E}[\hat{g}(X)]$$

Cette écriture peut alors être généralisée à tout modèle à l'aide de la valeur de Shapley.

6.2.3.4 *Valeur de Shapley dans le cas général*

On se place dans le cadre d'un modèle quelconque, avec $\hat{g}$ la fonction associée. on note $X = X_1$ x X_2 x ... x X_p l'espace des caractéristiques, représenté sur l'ensemble $\{1, ..., p\}$.

Considérons $\tilde{x} = (\tilde{x_1}, ..., \tilde{x_p})$ l'instance pour laquelle on veut expliquer la prédiction.

Définissons la différence en prédicition d'un sous-ensemble des valeurs des caractéristiques dans une instance particulière $\tilde{x}$, introduite par Strumbelj et Kononenko. Il s'agit du changement dans la prédiction causé par l'observation de ces valeurs des variables explicatives. Formellement, soit $S = \{i_1, ..., i_s\} \subset \{1, ..., p\}$ un sous-ensemble des variables explicatives (avec $s \in \{1, ..., p\}$). Notons $\Delta^{\tilde{x}}$ la différence de prédiction, associée au sous-ensemble S:

$$\Delta^{\tilde{x}}(S) = \mathbb{E}[\hat{g}(X_1, ..., X_p)|X_{i_1} = x_{i_1}, ..., X_{i_s} = x_{i_s}] - \mathbb{E}[\hat{g}(X_1, ..., X_p)]$$

Cette différence de prédiction correspond à notre fonction de coût. Ainsi $(\{1, ..., p\}, \Delta^{\tilde{x}})$ forme un jeu de coalition tel qu'il est défini dans la partie précédente.

La contribution de la variable explicative $x_j,\ j \in \{1, ..., p\}$, est définie comme la valeur de Shapley de ce jeu de coopération $(\{1, ..., p\}, \Delta^{\tilde{x}})$:

$$\phi_j(\Delta^{\tilde{x}}) = \sum_{S \in S(\{x_1, ..., x_p\} \backslash \{x_j\})} \frac{|S|!(p - |S - 1|)!}{p!} (\Delta^{\tilde{x}}(S \cup \{x_j\}) - \Delta^{\tilde{x}}(S)) \qquad (35)$$

Dans cette formule : $S(\{x_1, ..., x_p\} \backslash \{x_j\})$ représente l'ensemble des permutations de cet ensemble. En utilisant la formule alternative équivalente, on a également :

$$\forall i \in \{1, ..., p\},\ \phi_i(\Delta^{\tilde{x}}) = \frac{1}{p!} \sum_{O \in Perm(P)} \Delta^{\tilde{x}}(Pre^i(O) \cup \{i\}) - \Delta^{\tilde{x}}(Pre^i(O)) \qquad (36)$$

Avec : $Perm(P)$ l'ensemble des permutations de $P = \{1, ..., p\}$.

Prenons un exemple simple pour comprendre comment la valeur de Shapley fonctionne. Considérons un jeu avec trois joueurs $\{1, 2, 3\}$. On compte alors $2^3 = 8$ sous-ensembles

S possibles, à savoir : $\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}$ et $\{1,2,3\}$. En utilisant la formule de l'équation 36, on obtient :

$$\phi_1 = \frac{1}{3}(v(\{1,2,3\}) - \{2,3\}) + \frac{1}{6}(v(\{1,2\}) - \{2\}) + \frac{1}{6}(v(\{1,3\}) - \{3\}) + \frac{1}{3}(v(\{1\}) - v(\emptyset))$$
$$\phi_2 = \frac{1}{3}(v(\{1,2,3\}) - \{1,3\}) + \frac{1}{6}(v(\{1,2\}) - \{1\}) + \frac{1}{6}(v(\{2,3\}) - \{3\}) + \frac{1}{3}(v(\{2\}) - v(\emptyset))$$
$$\phi_3 = \frac{1}{3}(v(\{1,2,3\}) - \{1,2\}) + \frac{1}{6}(v(\{1,3\}) - \{1\}) + \frac{1}{6}(v(\{2,3\}) - \{2\}) + \frac{1}{3}(v(\{3\}) - v(\emptyset))$$

En définissant le gain "non distribué" $\phi_0 = v(\emptyset)$, qui correspond au payoff fixé qui n'est pas associé aux actions des joueurs, la propriété d'additivité est bien respectée, à savoir : $\phi_0 + \phi_1 + \phi_2 + \phi_3 = v(\{1,2,3\})$.

Dans le cas général, on retrouve alors les propriétés vues précédemment à savoir :

- Efficacité : $\sum_{i=1}^{p} \phi_i(\Delta^{\tilde{x}}\text{à}) = \Delta^{\tilde{x}}(\{1,...,p\}) = \hat{g}(\tilde{x}) - \mathbb{E}[\hat{g}(X)]$. On retrouve alors la propriété que l'on a observé pour le modèle linéaire, à savoir que la somme des p contributions pour l'explication d'une observation est égale à la différence entre la prédiction faite par le modèle pour cette observation et la prédiction (moyenne) du modèle si on ne connaissait aucune information sur la valeur des variables explicatives $x_j,\ j \in \{1,...,p\}$.
- Symétrie : deux variables explicatives qui ont une influence identique sur la prédiction auront des valeurs de contribution identiques
- Facticité : une variable qui a une contribution de 0 n'aura aucune influence sur la prédiction.
- Additivité : Si le modèle que l'on utilise repose sur la moyenne de plusieurs modèles (comme les forêts aléatoires qui utilisent des arbres de décision) alors la contribution de ce modèle sera la moyenne des contributions de chaque modèle pris seul.

6.2.3.5 *Algorithme de calcul approché de la valeur de Shapley*

Le problème, en pratique, est le temps de calcul de cette valeur de Shapley, du fait de la complexité (croissante avec le nombre de variables et de modalités). En effet, pour ce faire, nous devons calculer toutes les coalitions possibles avec ou sans la variable que l'on souhaite expliquer: il s'agit alors d'une complexité exponentielle.

Pour remédier à ce problème, Strumbelj et Kononenko dans *Explaining prediction models and individual predictions with feature contributions* ont proposé une approximation s'appuyant sur des méthodes de simulation par Monte Carlo, à savoir : $\hat{\phi_j} = \frac{1}{M}\sum_{m=1}^{M} \hat{g}(x^m_{+j}) - \hat{g}(x^m_{-j})$, où : $j \in \{1,...,p\}$ le numéro de la variable qu'on souhaite expliquer, $M \in \mathbb{N}$ est le nombre d'itérations choisi, $\hat{g}(x^m_{+j})$ la prédiction pour le vecteur

$x = (x_1, ..., x_p)$ de p variables explicatives, mais avec un nombre aléatoire de caractéristiques remplacées par un point z aléatoire, excepté pour la valeur de la caractéristique j choisie. $\hat{g}(x^m_{-j})$ est quasiment identique à $\hat{g}(x^m_{+j})$ sauf que la valeur x^m_j est aussi prise à partir de l'échantillon de x.

On en déduit la procédure proposée par Strumbelj et Kononenko pour approcher la valeur de Shapley $\phi_j(\Delta^{\tilde{x}})$ associée à la variable x_j pour $j \in \{1, ..., p\}$ à l'aide de l'algorithme suivant :

- Entrée : modèle $\hat{g}$, l'instance $\tilde{x}$ qu'on cherche à expliquer, M le nombre d'itérations de l'algorithme
- $\phi_j = 0$
- pour i allant de 1 à M, faire :
  - choisir une permutation aléatoire $O \in Perm(P)$
  - choisir une instance $z = (z_1, ..., z_p)$ du dataset initial
  - $\forall k \in \{1, ..., p\}$, $x^+_k = \begin{cases} \tilde{x}_k \text{ si } k \in Pre^i(O) \cup \{j\} \\ z_k \text{ sinon} \end{cases}$
  - $\forall k \in \{1, ..., p\}, x^-_k = \begin{cases} \tilde{x}_k \text{ si } k \in Pre^i(O) \\ z_k \text{ sinon} \end{cases}$
  - $\phi_j = \phi_j + (\hat{g}(x^+) - \hat{g}(x^-))$
- Sortie : $\hat{\phi_j}^{(M)} = \frac{\phi_M}{M}$

Notons bien qu'à chaque itération, les calculs des termes $\hat{g}(x^+)$ et $\hat{g}(x^-)$ reposent sur des observations qui sont identiques à l'exception de la variable $\tilde{x}_j$. Ils sont construits en prenant l'instance z et en changeant la valeur de chaque variable apparaissant avant la j-ième variable dans l'ordre de la permutation O (pour x^- la valeur de $\tilde{x}_j$ est également changée) par la valeur des caractéristiques de l'instance pour laquelle on désire expliquer y.

6.2.3.6 *Propriétés et limites de SHAP*

Bien que SHAP soit également un modèle d'interprétation local, il diffère néanmoins de LIME : SHAP explique la différence entre la prédiction et la prédiction moyenne globale, tandis que LIME explique la différence entre la prédiction et une prédiction moyenne locale.

SHAP est la seule méthode d'interprétabilité, à ce jour, avec un fondement mathématique. En effet, la différence entre la prédiction et la prédiction moyenne est distribuée

de manière "juste" entre les différentes variables utilisées par le modèle, grâce à la propriété d'efficacité de la valeur de Shapley. Ceci n'est pas le cas de LIME, qui repose sur un principe qui semble cohérent mais n'a pas de justification mathématique. Dans le cadre du RGPD et du "droit à l'explication", SHAP pourrait ainsi être une méthode d'interprétabilité des modèles répondant à ces exigences.

La méthode SHAP fournit une explication de la prédiction faite par un modèle quelconque (aussi complexe soit-il) en attribuant une valeur de contribution à chaque variable utilisée ; contrairement à LIME qui renvoie une réponse plus concise, en pénalisant les modèles complexes. On peut alors considérer que SHAP réalise moins d'approximations que LIME et de ce fait fournit une explication plus précise.

Lorsque le modèle à interpréter est entraîné avec un grand nombre de variables, l'interprétation fournie par SHAP n'est pas parcimonieuse. SHAP renvoie effectivement autant de coefficients que de variables explicatives rendant parfois la lecture difficile.

Pour contourner ce problème, une adaptation de SHAP, appelée Kernel Shap (Linear LIME + Shapley Values) est proposée. L'idée est ainsi de relier les équations 32 (de LIME) et 35 (de SHAP). En choisissant judicieusement la fonction de coût J, la mesure de proximité $\Pi_{x'}$ et le terme de régularisation Ω, il est alors possible d'écrire la valeur de Shapley comme solution du problème d'optimisation posé par LIME dans l'équation 32. Cette combinaison permet alors de fournir des explications plus parcimonieuses.

Par ailleurs, SHAP dans sa version initiale suppose que les variables sont indépendantes. Une alternative a néanmoins été récemment proposée par Aas et al. (2019).

Enfin, on remarquera que SHAP ne fournit qu'une indication sur la contribution de chaque variable pour une prédiction donnée. Il ne permet pas de déduire des effets globaux comme l'interprétation des *odd-ratios* dans le cadre de régression linéaire. Il n'apporte qu'une compréhension locale même si cette dernière est parfois plus explicite lors de l'usage de modèles complexes comme les réseaux de neurones ou des méthodes par arbres.

Ce chapitre nous a ainsi permis d'effectuer une revue de littérature des avancées récentes en matière d'interprétation des modèles. Le sujet de l'interprétabilité des modèles reste un enjeu majeur pour les industries. Pour les compagnies d'assurance ce thème est au cœur de la préoccupation des régulateurs dans un contexte où les nouvelles technologies amènent nécessairement l'usage d'algorithmes nouveaux pour la discipline. Ainsi, la lecture des différentes méthodes nous aura permis de s'informer sur les tenants et aboutissants des dernières publications et ainsi mieux appréhender ces problématiques.

7

CONCLUSION ET PERSPECTIVES

Le *machine learning* est une discipline qui ne cesse d'évoluer et ce de manière frénétique. Pouvant s'appliquer à toutes les industries, elle facilite l'échange des connaissances. Elle ouvre également de nombreux nouveaux axes de recherche. Éclairer les problématiques assurantielles sous le regard de l'apprentissage statistique nous aura permis de découvrir non seulement le potentiel de certains modèles de *machine learning*, mais également de mettre en avant les limites et points d'attention dans l'utilisation de ces derniers.

Plus généralement la science des données semble indissociable de la communauté et de ses valeurs. Parmi elles, le partage des découvertes et la collaboration sont sans doute les plus notables. Elles expliquent la rapidité avec laquelle ces dernières années la recherche a pu avancer et s'appliquer à toutes les industries de service. Elles nous ont notamment permis de transposer certaines techniques au monde de l'assurance et de les observer parfois sous un regard actuariel. Il n'est pas rare d'utiliser en parallèle des recherches en *machine learning* systématiquement un logiciel de gestion de versions comme l'outil *git* afin de pouvoir accéder à l'implémentation associée aux différentes publications. Ces services permettent notamment de partager librement les codes ayant produit les résultats des différents travaux [1] et surtout laissent à la communauté la possibilité d'amélioration ou de correction de ces codes. Le partage permet de favoriser les connexions entre les différents secteurs industriels mais surtout d'intensifier les échanges entre les écoles, laboratoires et les entreprises.

Cette thèse aura été pour moi l'occasion de contribuer -à mon échelle- à la jonction de ces différents univers. Même si ce manuscrit n'en fait pas nécessairement état, au-delà des différents sujets sur lesquels j'ai pu travailler, j'ai contribué à l'organisation de challenges internationaux : le Data Science Game[2] à destination des jeunes data scientists du monde entier. Ce challenge a été présenté lors d'un workshop de la conférence internationale NeurIPS [Ly et al. 2016]. Ce souci de vouloir relier les avancées de la recherche à l'industrie s'est par ailleurs traduit tout au long de ma thèse par mes travaux chez

1 Le site https://paperswithcode.com/ recense par exemple un grand nombre de papiers et des répertoires de code associés

2 datasciencegame.com

Milliman. La thèse retranscrit ainsi des sujets concrets que j'ai pu porter en tant que services proposés aux clients de Milliman.

La maîtrise des différents algorithmes s'est aussi accompagnée d'un apprentissage des techniques de programmation (Python, R, Shell Linux) et de développement de logiciels (CI, CD). Pouvoir transformer un algorithme en un service à forte valeur ajoutée nécessite en effet d'appréhender la complexité des services informatiques. Nous avons ainsi pu découvrir le calcul distribué et se familiariser notamment avec le framework Spark que nous avons déployé dans les bureaux de Milliman. Cette expérience est aujourd'hui un fort atout afin de comprendre l'intégralité des enjeux de l'application du *machine learning* dans le secteur de l'assurance. Elle nous a permis par exemple de contribuer largement à l'étude menée par la SOA [Ly 2019] mais également de travailler sur des sujets d'innovation comme l'utilisation des données télématiques en assurance automobile [Bellina and Ly 2018] bien que ces travaux ne soient pas détaillés ici.

Ces quatre années de thèse concluent ainsi mes débuts de la recherche appliquée à l'assurance. La plus grande difficulté restera sans doute l'assimilation des avancées toujours plus nombreuses et fréquentes dans ce domaine qui me passionne.

BIBLIOGRAPHY

[1] Le nouveau big-bang de l'actuariat. `http://www.argusdelassurance.com/acteurs/le-nouveau-big-bang-de-l-actuariat.67806`.

[2] K. Aas, M. Jullum, and A. Loland. Explaining individual predicitions when features are dependant : More accurate approximations to shapley values. *arXiv:1903.10464*, 2019.

[3] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6, 2018.

[4] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[5] C.C. Aggarwal. On k-anonymity and the curse of dimensionality. *IBM T. J. Watson Research Center*, 2005.

[6] Charu C. Aggarwal and Cheng Xian Zhai. A survey of text clustering algorithms. *Text Mining Data, Springer*, pages 77–121, 2012.

[7] Charu C. Aggarwal and Cheng Xian Zhai. An introduction to text mining. *Text Mining Data, Springer*, pages 4–10, 2012.

[8] I. Ahamada and E. Flachaire. Non-parametric econometrics. *Oxford University Press*, 2011.

[9] Lovell C.A.J Aigner, D. and P. Schmidt. Formulation and estimation of stochastic frontier production function models. *Journal of Econometrics*, 6:21–37, 1977.

[10] J. Aldrich. The econometricians' statisticians, 1895-1945. *History of Political Economy*, 42:111–154, 2010.

[11] Marco G. Altman, E. and F. Varetto. Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the italian experience). *Journal of Banking and Finance*, 18:505–529, 1994.

[12] J.D. Angrist and A.B. Krueger. Does compulsory school attendance affect schooling and earnings? *Quarterly Journal of Economics*, 106:979–1014, 1991.

[13] J.D. Angrist and V. Lavy. Using maimonides' rule to estimate the effect of class size on scholastic achievement. *Quarterly Journal of Economics*, 114:533–575, 1999.

[14] J.D. Angrist and J.S. Pischke. The credibility revolution in empirical economics: How better research design is taking the con out of econometrics. *Journal of Economic Perspective*, 24:3–30, 2010.

[15] J.D. Angrist and J.S Pischke. Mastering metrics. *Princeton University Presse*, 2015.

[16] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[17] Jean-Marc Aouizerate. Alternative neuronale en tarification santé. *Ressources Actuarielles*, 2010.

[18] D.W Apley. Visualizing the effects of predictor variables in black box supervised learning models. *arXiv:1612.08468*, 2014.

[19] D.W Apley. *Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models*. 2016.

[20] L. J. M. Aslett, M. Pedro Esperan, and Chris C. Holme. Encrypted statistical machine learning: new privacy preserving methods. 2015.

[21] N. Parmar J. Uszkoreit L. Jones A. N. Gomez L. Kaiser I. Polosukhin Aswani, N. Shazeer. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 12 2018.

[22] Nekipelov D. Ryan S.P. Bajari, P. and M. Yang. Machine learning methods for demand estimation. *American Economic Review*, 105:481–485, 2015.

[23] Michel Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Owen Etzioni. Open information extraction from the web. *20th Joint International Conference On Artificial Intelligence*, pages 2670–2676, 2007.

[24] R.J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. *Conference: Data Engineering*, 2005.

[25] S. Bazen and K. Charni. Do earnings really decline for older workers? *AMSE 2015-11 Discussion Paper, Aix-Marseille University*, 2015.

[26] Remi Bellina and Antoine Ly. A european insurance leader works with milliman to process raw telematics data and detect driving behaviors. *White paper, http://www.qa.milliman.com*, 2018.

[27] R.E Bellman. Dynamic programming. *Princeton University Press*, 1957.

[28] Chen D. Chernozhukov V. Belloni, A. and C. Hansen. Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica*, 80: 2369–2429, 2012.

[29] Chernozhukov V. Belloni, A. and C. Hansen. Inference methods for high-dimensional sparse econometric models. *Advances in Economics and Econometrics*, pages 245–295, 2010.

[30] Y. Bengio and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[31] Yoshua Bengio. A connectionist approach to speech recognition. *International journal of pattern recognition and artificial intelligence*, 7(04):647–667, 1993.

[32] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[33] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289–300, 1995.

[34] J.O. Berger. Statistical decision theory and bayesian analysis (2nd ed.). *Springer-Verlag*, 1985.

[35] R.A. Berk. Statistical learning from a regression perspective. *Springer Verlag*, 2008.

[36] J. Berkson. Applications of the logistic function to bioassay. *Journal of the American Statistical Association*, 9:357–365, 1944.

[37] J. Berkson. Why i prefer logits to probits. *Biometrics*, 7:327–339, 1951.

[38] J.M. Bernardo and A.F.M. Smith. Bayesian theory. *John Wiley*, 2000.

[39] E. R. Berndt. The practice of econometrics: Classic and contemporary. *Addison Wesley*, 1990.

[40] Gotze F. Bickel, P.J. and W. van Zwet. Resampling fewer than n observations: gains, losses and remedies for losses. *Statistica Sinica*, 7:1–31., 1997.

[41] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. *5th Conference on Applied Natural Language Processing*, page 194–201, 1997.

[42] C. Bishop. Pattern recognition and machine learning. *Springer Verlag*, 2006.

[43] J. Blanc and De Georges; A. Techniques de cryptographies. 2004.

[44] M. Lara J. Blanco, A. Pino-Mejias and S. Rayo. Credit scoring models for the microfinance industry using neural networks: Evidence from peru. *Expert Systems with Applications*, 40:356–364, 2013.

[45] C.I. Bliss. The method of probits. *Science*, 79:38–39, 1934.

[46] Ehrenfeucht-A. Haussler D. Blumer, A. and M.K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36:4:929–965, 1989.

[47] L. Bottou. Large-scale machine learning with stochastic gradient descent. *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, 2010.

[48] Hervé Bourlard and Nelson Morgan. Connectionist speech recognition-a hybrid approach. Technical report, KLUWER ACADEMIC PUBLISHERS, 1994.

[49] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16:199–231, 2001a.

[50] L. Breiman. Random forests. *Machine learning*, 45:5–32, 2001b.

[51] L.D. Brown. Fundamentals of statistical exponential families: with applications in statistical decision theory. *Institute of Mathematical Statistics, Hayworth*, 1986.

[52] P. Bühlmann and S. van de Geer. Statistics for high dimensional data: Methods, theory and applications. *Springer Verlag*, 2011.

[53] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. *16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, page 328–334, 1999.

[54] E. Candès and Y. Plan. Near-ideal model selection by ℓ_1 minimization. *The Annals of Statistics*, 37:2145–2177, 2009.

[55] G. Casiez. Cours analyse en composantes connexes.

[56] CEA. Solvency 2 glossary. *Groupe Consultatif Actuariel Européen*, 2008.

[57] Nicolo Cesa-Bianchi and Gabor Lugosi. Prediction learning and game. *Cambridge*, 2006.

[58] Arthur Charpentier, Emmanuel Flachaire, and Antoine Ly. Econometrics and machine learning. *Economics and Statistics*, 505-506, Big Data and Statistics: 147–169, 2019.

[59] Muller S. Charron C. « la réputation », mémoire réalisé dans le cadre du séminaire gestion des risques et environnement, master 2 innovation et management des technologies, université paris 1 panthéon sorbonne. 2011.

[60] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[61] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, and Ching Suen. *Character recognition systems: a guide for students and practitioners.* 2007.

[62] H. Civel. Modélisation du comportement client en assurance des emprunteurs sur le rachat anticipé de crédit. *Ressources Actuarielles*, 2015.

[63] Fokoué E. Clarke, B.S. and H.H. Zhang. Principles and theory for data mining and machine learning. *Springer Verlag*, 2009.

[64] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression.* New York: Chapman and Hall, 1982.

[65] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–292, 1995.

[66] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.

[67] T.M. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[68] T.M. Cover and J. Thomas. Elements of information theory. *Wiley*, 1991.

[69] G. Cybenko. Approximation by superpositions of a sigmoidal function 1989. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[70] J. L. Ba D. P. Kingma. *ADAM : A Method For Stochastic Optimization.* 2017.

[71] A. S. Dalayan. *Cours d'Apprentissage et Data Mining (Ensae PariTech).*

[72] Arnak Dalayan. Apprentissage statistique et datamining. *Cours ENSAE ParisTech - Datascience*, 2014.

[73] Luciana Dalla Valle, Dean Fantazzini, and Paolo Giudici. Copulae and operational risks.

[74] Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

[75] G. Darmois. Sur les lois de probabilites a estimation exhaustive. *Comptes Rendus de l'Académie des Sciencs, Paris*, 200:1265–1266, 1935.

[76] M. Daubechies, I. Defrise and C. De Mol. An iterative thresholding algorithm for linear inverse problems with sparsity constraint. *Communications on Pure and Applied Mathematics*, 57:11:1413–1457, 2004.

[77] R. Davidson and J.G. MacKinnon. Estimation and inference in econometrics. *Oxford University Press*, 1993.

[78] R. Davidson and J.G. MacKinnon. Econometric theory and methods. *Oxford University Press*, 2003.

[79] A.C. Davison. Bootstrap. *Cambridge University Press*, 1997.

[80] G. Debreu. Theoretic models: Mathematical form and economic content. *Oxford University Press*, 54:1259–1270, 1986.

[81] Gerald Dejong. Prediction and substantiation: a new approach to natural language processing. *Cognitive Science*, 3:251–273, 1979.

[82] Laurent Devineau and Matthieu Chauvigny. Replicating portfolios : techniques de calibrage pour le calcul du capital économique solvabilité 2. *HAL - Archives ouvertes*, hal-00508517, 2010.

[83] Laurent Devineau and Stéphane Loisel. Construction d'un algorithme d'accélération de la méthode des "simulations dans les simulations" pour le calcul du capital économique solvabilité ii. *Bulletin Francais d'Actuariat*, hal-00365363v2: 188–221, 2009.

[84] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[85] Jacob Devlin, Ming-Wei Chang, Lee Kenton, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv.org*, 2019.

[86] X. d'Haultefœuille and P. Givord. La régression quantile en pratique. *Economie and Statistiques*, 471:85–111, 2014.

[87] P. Dhillon, Y. Lu, D.P. Foster, and L.H. Ungar. New subsampling algorithms for fast least squares regression. *Advances in Neural Information Processing Systems*, 26, 2014.

[88] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. *Microsoft Research*, (MSR-TR-2016-3), February 2016.

[89] John Duchi, Elad Hazan, and Yoram Singer. Adaptative subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning*, 12: 2121–2159, 2011.

[90] Delbert Dueck and Brendan J Frey. Non-metric affinity propagation for unsupervised image categorization. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[91] Q. Duo. The formation of econometrics. *Oxford University Press*, 1993.

[92] Xavier Dupré. *Contributions à la reconnaissance de l'écriture cursive à l'aide des modèles de Markov cachès.* PhD thesis, Paris 5, 2004.

[93] J. Eder. Privacy in biobanks k-anonymity and l-diversity, etc. *Université de Klagenfurt*, 2012.

[94] B. Efron and R. Tibshirani. Bootstrap. *Chapman Hall CR*, 1993.

[95] EIOPA. Consultation paper on the proposal for guidelines on the use of internal models. 2014.

[96] IL Elmhurst. Optical character recognition and the years ahead, 1969.

[97] E. Engel. Die productions- und consumtionsverhältnisse des königreichs sachsen. *Statistisches Bureau des Königlich Sächsischen Ministeriums des Innern*, 1857.

[98] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[99] Stéphanie Fauvel and Maryse Le Pévédic. Analyse des rachats d'un portefeuille vie individuelle : Approche théorique et application pratique. *Ressources Actuarielles*, 2007.

[100] M. Feldstein and C. Horioka. Domestic saving and international capital flows. *Economic Journal*, 90:314–329, 1857.

[101] A. Fisher. *All Models are Wrong but Many are Useful: Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance.* 2018.

[102] P. Flach. Machine learning. *Cambridge University Press.*, 2012.

[103] D.P. Foster and E.I. George. The risk inflation criterion for multiple regression. *The Annals of Statistics*, 22:4:1947–1975, 1994.

[104] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[105] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, 1996.

[106] J.H. Friedman. Data mining and statistics: What's the connection. *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*, 1997.

[107] J.H. Friedman. Greedy function approximation : A gradient boosting machine. *The Annals of Statistics*, 2001.

[108] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[109] R. Frisch and F.V. Waugh. Partial time regressions as compared with individual trends. *Econometrica*, 1:387–401, 1933.

[110] Jean-Baptiste Garnier and Anne-Claire Martial. Optimisation du calcul du capital économique d'une compagnie d'assurance vie par la méthode des simulations dans les simulations. *Ressources Actuarielles*, 2013.

[111] Christophe Giraud. Analyse en composantes principales. *http://www.cmap.polytechnique.fr/~giraud/MSV/ACP.pdf*.

[112] P. Givord. Méthodes économétriques pour l'évaluation de politiques publiques. *INSEE Document de Travail*, 08, 2010.

[113] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *JMLR*, 9, 2010.

[114] T. Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106:746–762, 2011.

[115] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[116] A. Goldstein. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *arXiv:1309.6392*, 2014.

[117] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. 2016.

[118] Robert B. Gramacy and Mike Ludkovski. Sequential design for optimal stopping problems. arXiv:1309.3832v2 [q-fin.CP], 2014.

[119] Mariéthoz J. Grandvalet, Y. and S. Bengio. Interpretation of svms with an application to unbalanced classification. *Advances in Neural Information Processing Systems*, 18, 2005.

[120] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. *In Advances in neural information processing systems*, page 545–552, 2010.

[121] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[122] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[123] Alex Graves, Schmidhuber, and Jürgen. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems*, pages 545–552, 2009.

[124] Alex Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

[125] B. M. Greenwell. A simple and effective model-based variable importance measure. *arXiv:1805.04755*, 2018.

[126] Ralph Grishman and Beth Sundheim. Message understanding conference-6, a brief history. *16th International Conference of Computational Linguistics*, pages 466–471, 1996.

[127] Emmanuèle Grosicki and Haikal El Abed. Icdar 2009 handwriting recognition competition. pages 1398–1402, 2009.

[128] T. Groves and T. Rothenberg. A note on the expected value of an inverse matrix. *Biometrika*, 56:690–691, 1969.

[129] Antoine Guillot. Modélisation de comportements d'arbitrage par apprentissage statistique. *Rapport de Stage Long -ENSAE ParisTech*, 2014.

[130] S. Gupta, A. Nenkova, and D. Jurafsky. Measuring importance and query relevance in topic-focused multi-document summarization. *Association for Computational Linguistics*, pages 193–196, 2007.

[131] T. Haavelmo. The probability approach in econometrics. *Econometrica*, 12:1–115, 1944.

[132] Isaac Haik. Text mining et reconnaissance d'écriture appliqués à l'assurance. *SCOR Prix Actuariat*, 2018.

[133] T. Hastie and R. Tibshirani. Generalized additive models. *Chapman and Hall/CRC*, 1990.

[134] Tibshirani R. Hastie, T. and J. Friedman. The elements of statistical learning. *Springer Verlag*, 2009.

[135] Tibshirani W. Hastie, T. and M. Wainwright. Statistical learning with sparsity. *Chapman CRC*, 2015.

[136] Tibshiriani R. Hastie, T. and R.J. Tibshiriani. Extended comparisons of best subset selection, forward stepwise selection and the lasso. *ArXiV*, 2016.

[137] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. *Springer Series in Statistics*, pages 398–410, 2008.

[138] D.O. Hebb. The organization of behavior. *New York, Wiley*, 1949.

[139] J.J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47: 153–161, 1979.

[140] Tobias J.L. Heckman, J.J. and E. Vytlacil. Simple estimators for treatment parameters in a latent-variable framework. *Econometrica*, 85:748–755, 2003.

[141] D F. Hendry and H.-M. Krolzig. Automatic econometric model selection. *Timberlake Press*, 2001.

[142] Keilbach M. Graepel T. Bollmann-Sdorra P. Herbrich, R. and K. Obermayer. Neural networks in economics. computational techniques for modelling in economics. *Springer Verlag*, pages 169–196, 1999.

[143] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *American Association for the Advancement of Science*, 313:504–507, 2006.

[144] A.E. Hoerl. Applications of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.

[145] A.E. Hoerl and R.W. Kennard. Ridge regression: biased estimation for nonorthogonal problems . *This Week's Citation Classic*, 1981.

[146] P. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81:945–960, 1986.

[147] Hull. Options, futures, and other derivatives. *Pearson*, 8 Edition, 2011.

[148] Koehler A.B. Ord J.K. Hyndman, R. and R.D. Snyder. Forecasting with exponential smoothing. *Springer Verlag*, 2009.

[149] C. Szegedy I. J. Goodfellow, J. Shlens. *Explaining and Harnessing Adversial Examples*. 2015.

[150] O. Baret J. C. Simon. Cursive words recognition, from pixel to features iii : Frontiers in handwriting recognition. *Elsevier Science Publishers B.V.*, pages 241–260, 1992.

[151] D. Witten T. Hastie James, G. and R. Tibshirani. An introduction to statistical learning. *Springer Series in Statistics*, 2013.

[152] Jing Jiang. *Information Extraction from Text*, pages 11–41. 2012.

[153] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. *Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, page 113–120, 2007.

[154] Michael Irwin Jordan. *Learning in graphical models*, volume 89. Springer Science & Business Media, 1998.

[155] Dan Jurafsky. Text classification and naïve bayes. *Stanford, https://web.stanford.edu/class/cs124/lec/naivebayes.pdf*, 2011.

[156] M.P. Kean. Structural vs. atheoretic approaches to econometrics.. *Journal of Econometrics*, 156:3–20, 2010.

[157] A. Khashman. Credit risk evaluation using neural networks: Emotional versus conven- tional models. *Applied Soft Computing*, 11:5477–5484, 2011.

[158] Mohammad S Khorsheed. Recognising handwritten arabic manuscripts using a single hidden markov model. *Pattern Recognition Letters*, 24(14):2235–2242, 2003.

[159] B. Kim. *Examples are not Enough, Learn to Criticize! Criticism for Interpretability.* 2016.

[160] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[161] F. Kleber, S. Fiel, M. Diem, and R. Sablatnig. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. pages 560–564, 2013.

[162] Talwalkar A. Sarkar P. Kleiner, A. and M. Jordan. The big data bootstrap. *arXiv*, 2012.

[163] S. Knerr, Y. Tay, P. Lallican, M. Khalid, and C. Viard-Gaudin. An offline cursive handwritten word recognition system. *In Proceedings of IEEE Region 10 Conference*, 2001.

[164] I. Koch. Analysis of multivariate and high-dimensional data. *Cambridge University Press*, 2013.

[165] R. Koenker. Quantile regression. *Cambridge University Press*, 2003.

[166] R. Koenker and J. Machado. Goodness of fit and related inference processes for quantile regression. *Journal of the American Statistical Association*, 94:1296–1309, 1999.

[167] R. Koenker, Edgeworth Galton, and J. Frish. Prospects for quantile regression in econometrics. *Conference on Principles of Econometrics*, 1998.

[168] P. W. Koh. *Understanding Black-box Predictions via Influence Functions.* 2017.

[169] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.

[170] T.C. Koopmans. Three essays on the state of economic science. *McGraw-Hill*, 1957.

[171] Adam Koursaris. A least squares monte carlo approach to liability proxy modelling and capital calculation. *Insurance Risk and Capital - Barrie Hibbert*, 2011.

[172] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[173] M. Kuhn and K. Johnson. Applied predictive modeling. *Springer Verlag*, 2013.

[174] J.R. Landis and G.G. Koch. The measurement of observer agreement for categorical data. *Springer Verlag*, 33:159–174, 1977.

[175] F. Lange. *Exploration de la valeur de Shapley et des indices d'interaction pour les jeux définis sur des ensembles ordonnés.* 2008.

[176] T. Laugel. *Inverse Classification for Comparison-based Interpretability in Machine Learning.* 2017.

[177] N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. *Technical Report 00674995, HAL*, 2012.

[178] E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18:690–706, 1996.

[179] Yann Lecun and Yoshua Bengio. Convolutional networks for images, speech, and time-series. *researchgate*, 1995.

[180] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[181] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436, 2015.

[182] H. Leeb. Evaluation and selection of models for out-of-sample prediction when the sample size is small relative to the complexity of the data-generating process. *Bernoulli*, 14:661–690, 2008.

[183] Françoise Legardeur. Comité scientifique pour l'histoire de l'assurance. *Guide des sources sur l'histoire de l'assurance*, 2-912916-91-7, 2007.

[184] T. Lemieux. The « mincer equation » thirty years after schooling, experience, and earnings. *Jacob Mincer A Pioneer of Modern Labor Economics, Grossbard Eds*, 14:127–145, 2006.

[185] J. Li and J. S. Racine. Nonparametric econometrics. *Princeton University Press*, 2006.

[186] Li Q. Racine J. Li, C. and D. Zhang. Optimal model averaging of varying coefficient models. *Department of Economics Working Papers 2017-01*, 2017.

[187] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[188] Tegmark M. Lin, H.W. and D. Rolnick. Why does deep and cheap learning work so well? *ArXiv e-prints*, 2016.

[189] Z. C. Lipton. The mythos of model interpretability. *arXiv.org:1606.03490*, 2017.

[190] Marcus Liwicki, Alex Graves, Horst Bunke, and Jürgen Schmidhuber. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 367–371, 2007.

[191] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[192] Olivier Lopez, Xavier Milhaud, and Pierre-E Thérond. Tree-based censored regression with application to insurance. *HAL*, hal-01141228, 2015.

[193] R.E. Lucas. Econometric policy evaluation: A critique. *Carnegie-Rochester Conference Series on Public Policy*, pages 19–46, 1976.

[194] S. M Lundberg and S. Lee. A unified approach to interpreting model predictions. *arXiv:1705.07874*, 2017.

[195] Antoine Ly. Etude de risques comportementaux et détermination du capital économique par la méthode des réseaux de neurones. *Ressources actuarielles*, 2015.

[196] Antoine Ly and Thomas Poinsignon. Non life pricing procedure on encrypted and anonymous data at the time of the gdpr. *White paper milliman.com,* ***article in progess with Thomas Poinsignon and Romuald Elie***, 2019.

[197] Antoine Ly, Antoine Miech, Thibault Laugel, and Benjamin Donnot. Data science game. *NIPS workshop - CiML - Poster*, 2016.

[198] Antoine et al. Ly. Cloud computing and machine learning uses in the actuarial profession. *SOA, Innovation and Technology*, 2019.

[199] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE, 2006.

[200] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

[201] C.L. Mallows. Some comments on c_p. *Technometrics*, 15:661–675, 1973.

[202] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[203] W.S. McCullogh and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:4:115–133, 1943.

[204] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[205] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[206] Xavier Milhaud. Segmentation et modélisation des comportements de rachat en assurance vie. *Ressources Actuarielles*, 2011.

[207] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *arXiv:1706.07269*, 2017.

[208] J. Mincer. Schooling, experience and earnings. *Columbia University Press*, 1974.

[209] T. Mitchell. Machine learning. *McGraw-Hill*, 1997.

[210] Mehryar Mohri, Afshin Rostaminazed, and Ameet Talwalkar. Foundation of machine learning. *MIT Press*, 2012.

[211] Rostamizadeh A. Mohri, M. and A. Talwalker. Foundations of machine learning. *MIT Press*, 2012.

[212] C. Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. 2019.

[213] J.N. Morgan and J.A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistcal Association*, 58:415–434, 1963.

[214] M.S. Morgan. The history of econometric ideas. *Cambridge University Press*, 1990.

[215] S. Mullainathan and J. Spiess. Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, 31:87–106, 2017.

[216] W. J Murdoch and C. Singh. Interpretable machine learning: Definitions, methods, and applications. *arXiv:1901.04592*, 2019.

[217] K. M. Murphy and F. Welch. Empirical age-earnings profiles. *Journal of Labor Economics*, 8:202–229, 1990.

[218] K.R. Murphy. Machine learning: a probabilistic perspective. *MIT Press*, 2012.

[219] M. Müller. Generalized linear models. *Handbook of Computational Statistics, Springer Verlag*, 2011.

[220] E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–200, 1964.

[221] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large datasets (how to break anonymity of the netflix prize dataset). *University of Texas at Austin*, 2008.

[222] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227—-234, 1995.

[223] Ani Nenkova and Kathleen McKeown. *A Survey of Text Summarization Techniques*. 2012.

[224] A. Nevo and M.D. Whinston. Taking the dogma out of econometrics: Structural modeling and credible inference. *Journal of Economic Perspective*, 24:69–82, 2010.

[225] J. Neyman. Sur les applications de la théorie des probabilités aux expériences agricoles : Essai des principes. mémoire de master. *Statistical Science*, 5:463–472, 1923.

[226] B. Nguyen. Techniques d'anonymisation. *Statistique et société*, 2, 2014.

[227] Marine Niedzwiedz. Utilisation des supports vecteurs machines pour l'accélération du calcul du capital économique. *Ressources Actuarielles*, 2014.

[228] Elder J. Nisbet, R. and G. Miner. Handbook of statistical analysis and data mining. *Academic Press*, 2011.

[229] A. Okun. Potential gnp: Its measurement and significance. *Proceedings of the Business and Economics Section of the American Statistical Association*, pages 98–103, 1962.

[230] Julian D Olden, Michael K Joy, and Russell G Death. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, 178(3-4):389–397, 2004.

[231] G.H. Orcutt. Toward a partial redirection of econometrics. *Review of Economics and Statistics*, 34:195–213, 1952.

[232] Wanli Ouyang, Xiao Chu, and Xiaogang Wang. Multi-source deep learning for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2329–2336, 2014.

[233] J. D Owens, M. Houston, D. Luebke, S. Green, J. E Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96:879–899, 2008.

[234] Boistel P. Le management de la réputation chez sernam : application du modèle ips. 2007.

[235] M. Telgarsky P. L. Bartlett, D.J Foster. *Spectrally-normalized margin bounds for neural networks*. 2017.

[236] A. Pagan and A. Ullah. Nonparametric econometrics. themes in modern econometrics. *Cambridge: Cambridge University Press*, 1999.

[237] Marc Parizeau. Réseaux de neurones. *Manuel de cours de l'Université de Laval*, 2004.

[238] C. Patvardhan, A. K. Verma, and C. V. Lakshmi. Denoising of document images using discrete curvelet transform for ocr applications. *International Journal of Computer Applications*, 55:0975 – 8887, 2012.

[239] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559—572, 1901.

[240] K. Pearson. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *dvances in Large Margin Classifiers*, 10:61–74, 1999.

[241] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[242] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE, 2014.

[243] Fernando J. Pineda. Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett.*, 59:2229–2232, 1987.

[244] Thomas Poinsignon. Processus de tarification non-vie sur des données chiffrées & anonymisées. *Ressources Actuarielles*, 2018.

[245] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[246] S. Portnoy. Asymptotic behavior of likelihood methods for exponential families when the number of parameters tends to infinity. *Annals of Statistics*, 16:356–366, 1988.

[247] Princeton. Auerbach on optical character recognition, 1971.

[248] M. H. Quenouille. Problems in plane sampling. *The Annals of Mathematical Statistics*, 20:355–375, 1949.

[249] M. H. Quenouille. Notes on bias in estimation. *Biometrika*, 43:353–360, 1956.

[250] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1956.

[251] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. 77:257–286, 1989.

[252] Lisa F. Rau. Extracting company names from text. *7th IEEE Conference on Artificial Intelligence Application*, page 29–32, 1991.

[253] O. Reiersø l. Confluence analysis of means of instrumental sets of variables. *Arkiv. for Mathematik, Astronomi Och Fysik*, 32:81–106, 1945.

[254] M. T. Ribeiro. *Anchors - High-Precision Model-Agnostic Explanations*. 2018.

[255] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" - explaining the predictions of any classifier. *arXiv:1602.049388*, 2016.

[256] Anne-Cécile Richard. La garantie en capital dans les produits d'épargne : de nouvelles solutions pour les assureurs. *Ressources Actuarielles*, 2013.

[257] P. Rosenbaum and D. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70:41–55, 1983.

[258] Frank Rosenblatt. Principles of neurodynamics. perceptron and the theory of brain mecanisms. 1960.

[259] D. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66:688–7018, 1974.

[260] Dong Rui and David A. Smith. Multi-input attention for unsupervised ocr correction. *ACL*, 2018.

[261] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[262] Wand M. P. Ruppert, D. and R.J. Carroll. Semiparametric regression. *Cambridge University Press*, 2003.

[263] A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44, 1959.

[264] Marc Sauget. Parallélisation de problèmes d'apprentissage par des réseaux neuronaux artificiels. application en radiothérapie externe. *HAL*, 2008.

[265] H. F Schantz. *The history of OCR, optical character recognition.* Manchester Center,Vt.: Recognition Technologies Users Association, 1982.

[266] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[267] Robert E. Schapire and Yoav Freund. Boosting. Foundations and Algorithms, 2015.

[268] H. Schultz. The meaning of statistical demand curves. *University of Chicago*, 1930.

[269] S.S. Shai and B.D. Shai. Understanding machine learning from theory to algorithms. *Cambridge University Press*, 2014.

[270] S. Shalev-Shwartz and S. Ben-David. Understanding machine learning: From theory to algorithms. *Cambridge University Press*, 2014.

[271] J. Shao. Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88:486–494, 1993.

[272] J. Shao. An asymptotic theory for linear model selection. *Statistica Sinica*, 7: 221–264, 1997.

[273] R.E. Shapire and Y. Freund. Boosting. *MIT Press*, 2012.

[274] B.W. Silverman. Density estimation. *Chapman and Hall*, 1986.

[275] J. S. Simonoff. Smoothing methods in statistics. *Springer*, 1996.

[276] K. Simonyan. *Deep Inside Convolutional Network - Visualising Image Classification Models and Saliency Maps*. 2014.

[277] Ray Smith. An overview of the tesseract ocr engine. *IEEE, Document Analysis and Recognition, 2007. ICDAR 2007. Nineth International Conference*, 2:629–633, 2007.

[278] Ray Smith, Daria Antonova, and Dar-Shyang Lee. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, page 1. ACM, 2009.

[279] Raymond W Smith. Hybrid page layout analysis via tab-stop detection. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 241–245. IEEE, 2009.

[280] Jan Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer Science & Business Media, 2005.

[281] Florent Pratlong Sophie Gaultier-Gaillard. Le risque de réputation : le cas du secteur bancaire. 2011.

[282] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. 2017. URL `http://arxiv.org/abs/1612.03975`.

[283] M. Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society. Series B*, 39:44–47, 1977.

[284] E. Strumbelj and I. Kononenko. A general method for visualizing and explaining black-box regression models. *Int. Conf. on Adaptive and Natural Computing Algorithms*, 2011.

[285] L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 2002.

[286] C. Szegedy. *Intriguing Properties of Neural Networks*. 2014.

[287] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[288] X. Renard T. Laugel. *Defining Locality for Surrogates in Post-hoc Interpretablity*. 2018.

[289] K.Y. Tam and M.Y. Kiang. Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, 38:926–947, 1992.

[290] H. Tan. Neural-network model for stock forecasting. *MSc Thesis, Texas Tech. University*, 1995.

[291] H. F. Tan, K. Song, and M. Udell. Why should you trust my interpretation? understanding uncertainty in lime predictions. *arXiv:1904.12991*, 2019.

[292] Huajin Tang, Kay Chen Tan, and Yi Zhang. Neural networks - computational models and applications. *Studies un Computational Intelligence*, 53, 2007.

[293] E. Teske-Wilson. Homomorphic cryptosystems. *University of Waterloo*, 2011.

[294] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[295] R. Tibshirani and L. Wasserman. A closer look at sparse regression. *http://bit.ly/2FrGQ32*, 2016.

[296] Robert John Tibshirani. *Generalized additive models*. London: Chapman and Hall, 1990.

[297] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics*, 4:1035–1038, 1963.

[298] J. Tinbergen. Statistical testing of business cycle theories. vol. 1: A method and its application to investment activity; vol. 2: Business cycles in the united states of america, 1919—1932. *Geneva: League of Nations*, 1939.

[299] J. Tobin. Estimation of relationship for limited dependent variables. *Econometrica*, 26:24–36, 1958.

[300] J. Tobin. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3:115–126, 2011.

[301] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[302] P. Tsen. Convergence of a block coordinate descent for nondifferentiable minization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001.

[303] S. Tufféry. Data mining and statistics for decision making. *Wiley Interscience*, 2001.

[304] J. W. Tukey. Bias and confidence in not quite large samples. *The Annals of Mathematical Statistics*, 29:614–623, 1958.

[305] L.G. Vailiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[306] C Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[307] V. Vapnik. Statistical learning theory. *Wiley*, 1998.

[308] H.R. Varian. Big data: New tricks for econometrics. *Journal of Economic Perspectives*, 28:3–28, 2014.

[309] F. Vermet. *Cours d'Apprentissage Statistique : une approche connexionniste (EURIA).*

[310] J.P. Vert. Machine learning in computational biology. *ENSAE*, 2017.

[311] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: a line segment detector. *Image Processing On Line*, 2:35–55, 2012.

[312] S. Wachter. *Counterfactual Explanations Without Opening the Black Box : Automated Decisions and the GDPR.* 2017.

[313] L.S. Waltrup, F. Sobotka, T. Kneib, and G. Kauermann. Expectile and quantile regression—david and goliath? *Statistical Modelling*, 15:433–456, 2014.

[314] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[315] G. S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics, Series A*, 26:359–372, 1964.

[316] Borhani R. Watt, J. and A. Katsaggelos. Machine learning refined : Foundations, algorithms, and applications. *Cambridge University Press*, 2016.

[317] H. White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464, 1989.

[318] J. M. White and G. D. Rohrer. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM Journal of Research and Development*, 27:400 – 411, 1983.

[319] B. Widrow and M.E. Hoff. Adaptative switching circuits. *Stanford Electronics Laboratories, Technical Report*, 1553, 1960.

[320] B. Widrow and M.E. Jr. Hoff. Adaptive switching circuits. *IRE WESCON Convention Record*, 4:96–104, 1960.

[321] Eyal Winter. The shapley value. *Handbook of game theory with economic applications*, 3:2025–2054, 2002.

[322] David Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, pages 1341–1390, 1997.

[323] Macready W.G. Wolpert, D.H. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67, 1997.

[324] E. J. Working. What do statistical 'demand curves' show? *Quarterly Journal of Economics*, 41:12–35, 1927.

[325] X. et al Yi. Homomorphic encryption and applications. *Springer*, 2014.

[326] Zhifeng Chen Quoc V. Le Mohammad Norouzi Wolfgang Macherey Maxim Krikun Yuan Cao Qin Gao KlausMacherey et al. Yonghui Wu, Mike Schuster. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[327] K. Yu and R. Moyeed. Bayesian quantile regression. *Statistics and Probability Letters*, 54:437–447, 2001.

[328] Dmitry Zelenko, Aone Chinatsu, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

[329] Min Zhang, Jie Zhang, and Jian Su. Exploring syntactic features for relation extraction using a convolution tree kernel. *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, page 288–295, 2006.

[330] M.A. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient. *Advances in neural information processing systems*, pages 2595–2603, 2001.

[331] Weimer Zinkevich, Smola, and Li. Parallelized stochastic gradient descent. *Advances in Neural Information Processing Systems 23*, pages 2595–2603, 2010.

8

TABLES AND FIGURES

LIST OF FIGURES

LIST OF TABLES

A

ANNEXES

Les annexes qui suivent peuvent parfois être lues indépendamment des différents chapitres de cette thèse. Elles constituent un complément d'information sur les différentes notions ou travaux qui ont pu être explorés lors de la thèse.

UNE PREMIÈRE EXPLORATION DE L'APPRENTISSAGE STATISTIQUE EN ACTUARIAT

A.1 LE CRITÈRE D'INFORMATION AIC ET BIC

Soit $(X_i, Y_i)_{i\in\mathbb{N}^*}$ une suite de variable i.i.d définie sur un même espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$. Pour tout $i \in \mathbb{N}^*$, Y_i est à valeur dans $\mathbb{R}$. On suppose qu'il existe $K \in \mathbb{N}^*$ tel que $(X_i)_{i\in\mathbb{N}^*}$ est à valeur dans $\mathcal{M}_{1,K}(\mathbb{R})$. Soit (X, Y) un couple de variables aléatoires définies sur $(\Omega, \mathcal{F}, \mathbb{P})$ avec la même loi de probabilité que $(X_i, Y_i)_{i\in\mathbb{N}^*}$.
Nous supposons qu'il existe $b \in \mathbb{M}_{K,1}(\mathbb{R})$ et $\sigma \in \mathbb{R}^{+*}$ tel que la densité de (X, Y) relativement à la mesure produit $\mu \otimes \lambda$ (μ étant la mesure sigma finie sur $\mathbb{R}^K$ et sa tribu borélienne, λ la mesure de Lebesgue sur $\mathbb{R}$) existe et s'écrit ainsi :

$$f_{(X,Y)}(x, y) = f_1(x) \times \frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left(-\frac{1}{2\sigma^2}(y - x.b)^2\right)$$

où f_1 correspond à la densité de X relativement à la mesure μ et est supposée indépendante du vecteur de paramètre b; où le terme de droite $(\frac{1}{\sqrt{2\pi\sigma^2}} \times \exp -\frac{1}{2\sigma^2}(y - x.b)^2)$ correspond à la densité conditionnelle de Y relativement à la mesure de Lebesgue λ sachant que $X = x$.

La vraissemblance des N premières observations, que l'on peut définir comme la densité relativement à la mesure produit $(\mu \otimes \lambda)^{\otimes N})$ définie sur $(\mathbb{R}^K \times \mathbb{R})^N$ du vecteur aléatoire $(X_{i,i})_{i\in[\![1,N]\!]}$ est donnée par :

$$\begin{aligned}\mathcal{L}((X_{i,i})_{i\in[\![1,N]\!]}, (b\sigma^2)) &= \prod_{i=1}^{N} f_{(X_i,Y_i)}(X_i, Y_i) \\ &= g((X_i)_{i\in[\![1,N]\!]}) \times \frac{1}{(2\pi\sigma^2)^{(\frac{N}{2})}} \times \exp(-\frac{1}{2\sigma^2}\|\mathbb{Y}^N - \mathbb{X}^N b\|_N^2)\end{aligned} \tag{37}$$

Où l'on note :

- $g((X_i)_{i\in[\![1,N]\!]}) := \prod_{i=1}^{N} f_1(X_i)$;
- $\|.\|_N$ désigne la norme euclidienne sur $\mathbb{R}^N$;

- $\mathbb{Y}^N$ est le vecteur colonne de $\mathbb{R}^N$ égal à $\begin{pmatrix} Y_1 \\ ... \\ Y_N \end{pmatrix}$;

- $\mathbb{X}^N$ est la matrice de $\mathcal{M}_{N,K}(\mathbb{R})$ telle que $\forall i \in [\![1, N]\!]$, la i-ème ligne de $\mathbb{X}^N$ est égale à X_i.

Remarque : la vraisemblance est une variable aléatoire définie sur $(\Omega, \mathcal{F}, \mathbb{P})$, fonction des paramètres $b \in \mathcal{M}_{K,1}(\mathbb{R})$ et $\sigma \in \mathbb{R}_+^*$.

Supposons que pour tout $N \in \mathbb{N}^*$ tel que $N \geq K$, la matrice aléatoire $\mathbb{X}^N$ est de rang maximal égal à K (sur Ω). Ainsi, la matrice aléatoire $\mathbb{X}^{N\prime}\mathbb{X}^N$ de $\mathcal{M}_K(\mathbb{R})$ prend ses valeurs dans un jeu de matrices inversibles de taille K. Les estimateurs du maximum de vraisemblance des paramètres $b \in \mathcal{M}_{K,1}(\mathbb{R})$ et $\sigma \in \mathbb{R}_+^*$ sont les variables aléatoires sur $(\Omega, \mathcal{F}, \mathbb{P})$ définies par :

$$(\hat{b}^N, \hat{\sigma^2}^N) := argmax[\mathcal{L}((X_i, Y_i)_{i\in[\![1,N]\!]}, (b, \sigma^2)), (b, \sigma^2) \in \mathbb{R} \times \mathbb{R}_+^*] \tag{38}$$

Proposition. Sous les hypothèse d'inversibilité ci-dessus, on a :

- $\hat{b}^N = (\mathbb{X}^{N\prime}\mathbb{X}^N)^{-1}\mathbb{X}^{N\prime}\mathbb{Y}^N$;

- $\hat{\sigma^2}^N = \frac{1}{N}\|\mathbb{Y}^N - \mathbb{X}^N\hat{b}^N\|_N^2$

Proof. Maximiser la vraisemblance pour les paramètres $(b, \sigma^2) \in \mathbb{R} \times \mathbb{R}_+^*$ revient à maximiser la log-vraisemblance en ces paramètres. La log-vraisemblance est notée $l((X_i, Y_i)_{i\in[\![1,N]\!]}, (b, \sigma^2)$ et vaut

$$l((X_i, Y_i)_{i\in[\![1,N]\!]}, (b, \sigma^2)) := \ln(g((X_i)_{i\in[\![1,N]\!]})) - \frac{N}{2}ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\|\mathbb{Y}^N - \mathbb{X}^N b\|_N^2 \tag{39}$$

On admet que les conditions de premier ordre assurent que les valeurs ainsi calculées donnent bien des maximums de la log-vraisemblance (ce qui peut être facilement montré grâce aux dérivées secondes). Les estimateurs $(\hat{b}^N, \hat{\sigma^2}^N)$ sont les solutions du système suivant :

$$\frac{\partial l((X_i, Y_i)_{i\in[\![1,N]\!]}, (b, \sigma^2))}{\partial b} = 0$$
$$\frac{\partial l((X_i, Y_i)_{i\in[\![1,N]\!]}, (b, \sigma^2))}{\partial \sigma^2} = 0$$

Ce système peut être réécrit

$$\mathbb{X}^{N\prime}(\mathbb{Y}^N - \mathbb{X}^N b) = 0_{\mathcal{M}_{K,1}(\mathbb{R})}$$
$$-\frac{N}{\sigma^2} + \frac{1}{(\sigma^2)^2}\|\mathbb{Y}^N - \mathbb{X}^N b\|_N^2 = 0_{\mathbb{R}}$$

On en déduit le résultat. Fin de la preuve. □

Le critère AIC du modèle linéaire gaussien d'ordre N et associé aux N observations $(X_i, Y_i)_{i\in[\![1,N]\!]}$ est donc donné par

$$AIC(N, (X_i, Y_i)_{i\in[\![1,N]\!]}) = -2 \times \mathcal{L}((X_i, Y_i)_{i\in[\![1,N]\!]}, (\hat{b}^N, \hat{\sigma^2}^N)|(X_i)_{i\in[\![1,N]\!]}) + 2 \times (K+1)$$

Où $K+1$ correspond au nombre de paramètres réels dans le modèle et $\mathcal{L}((X_i, Y_i)_{i\in[\![1,N]\!]}, (\hat{b}^N, \hat{\sigma^2}^N)|(X_i)_{i\in[\![1,N]\!]})$ est la vraisemblance conditionnelle du vecteur $(X_i, Y_i)_{i\in[\![1,N]\!]}$ sachant le vecteur d'observations $(X_i)_{i\in[\![1,N]\!]}$, donné par

$$\mathcal{L}((X_i, Y_i)_{i\in[\![1,N]\!]}, (\hat{b}^N, \hat{\sigma^2}^N)|(X_i)_{i\in[\![1,N]\!]}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \times \exp(-\frac{1}{2\sigma^2}\|\mathbb{Y}^N - \mathbb{X}^N b\|_N^2)$$

On vérifie aisément :

$$AIC(N, (X_i, Y_i)_{i\in[\![1,N]\!]}) = N \times (\ln(2\pi) + 1 + \ln(\hat{\sigma^2}^N)) + 2 \times (K+1)$$

De même, on obtient BIC :

$$BIC(N, (X_i, Y_i)_{i\in[\![1,N]\!]}) = -2 \times \mathcal{L}((X_i, Y_i)_{i\in[\![1,N]\!]}, (\hat{b}^N, \hat{\sigma^2}^N)|(X_i)_{i\in[\![1,N]\!]}) + \ln(N) \times (K+1)$$

A.2 RÉGRESSION STEPWISE

Afin de mieux comprendre l'algorithme de sélection des facteurs de risque (Stepwise), nous présentons sous forme pseudo-codé la méthode stepwise utilisée.

Soit :

- $D \in \mathbb{N}^*$ le nombre de variables explicatives ;
- $N \in \mathbb{N}^*$ le nombre d'observations;
- Pour chaque observation $i \in [\![1, N]\!]$ on note Y_i la variable à expliquer et X_i le vecteur ligne contenant les $d \leq D$ valeurs des variables explicatives;
- Pour tout $j \in [\![1, D]\!]$, on note X^j la $j^{ième}$ variable explicative.

- $X_{candidats} = \mathcal{P}(\left\{X^1, ..., X^D\right\}$ l'ensemble des variables explicatives candidates;
- $X_{select} = X^0 \cup \mathcal{P}(\left\{X^1, ..., X^D\right\}$ l'ensemble des variables explicatives retenues après une itération, avec X^0 représentant la constante (variable explicative par défaut);
- $AIC\_BIC(X_{select})$, le critère d'information calculé sur le jeu.
- $k_{max} \in [\![1, D]\!]$, l'indice de la variable choisi parmi $X_{candidats}$ tel que $k_{max} = \underset{j}{argmax} AIC\_BIC(X_{select} \cup \{X^j\})$

#INITIALISATION
$X_{select} = X^0$
$X_{candidats} = \{X^1, ..., X^D\}$
$critere_{old} = AIC\_BIC(X_{select})$
Compute k_{max} $critere_{new} = AIC\_BIC(X_{select} \cup \{X^{k_{max}}\})$

#BOUCLE DE TEST
While $critere_{new} < critere_{old}$ – la sélection s'effectue tant que le critère décroît
$critere_{old} = AIC\_BIC(X_{select})$
Compute k_{max}
$X_{select} = X_{select} \cup X^{k_{max}}$
$X_{candidats} = X_{candidats} - \{X^{k_{max}}\}$
$critere_{new} = AIC\_BIC(X_{select})$
EndWhile

#FIN
Return X_{select}

Une fois les facteurs de risque sélectionnés, une régression multi-linéaire simple permet de déterminer les différents coefficients associés aux facteurs retenus.

A.3 LES RÉSEAUX DE NEURONES RÉCURRENTS

Les réseaux de neurones récurrents (RNN) sont des architectures de couches de neurones permettant de traiter des données séquentielles avec des dépendances temporelles. Ces réseaux ont de plus la particularité de pouvoir considérer des vecteurs d'entrée de tailles différentes. Plus généralement, l'idée de ces réseaux est de répliquer la notion de mémoire et ainsi permettre la diffusion d'un signal tout au long d'une même couche. La comparaison d'un RNN et d'un réseau traditionnel est illustrée en figure 102 afin de mieux visualiser ce qui le diffère d'un réseau plus traditionnel.

On peut ainsi constater que le RNN introduit des connections entre les neurones d'une même couche.

Plus formellement si on note $x^{(1)}, ..., x^{(T)}$ le vecteur d'entrée du réseau, la sortie d'un neurone récurrent noté $h^{(t)}$ est alors définie par

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \theta)$$

avec θ l'ensemble des paramètres associés de la fonction f. La fonction f peut être identique pour chaque composante t ce qui assure le transfert des paramètres θ au sein du réseau. Plus concrètement, la fonction f est souvent modélisée par une fonction linéaire comme l'illustre la figure 103.

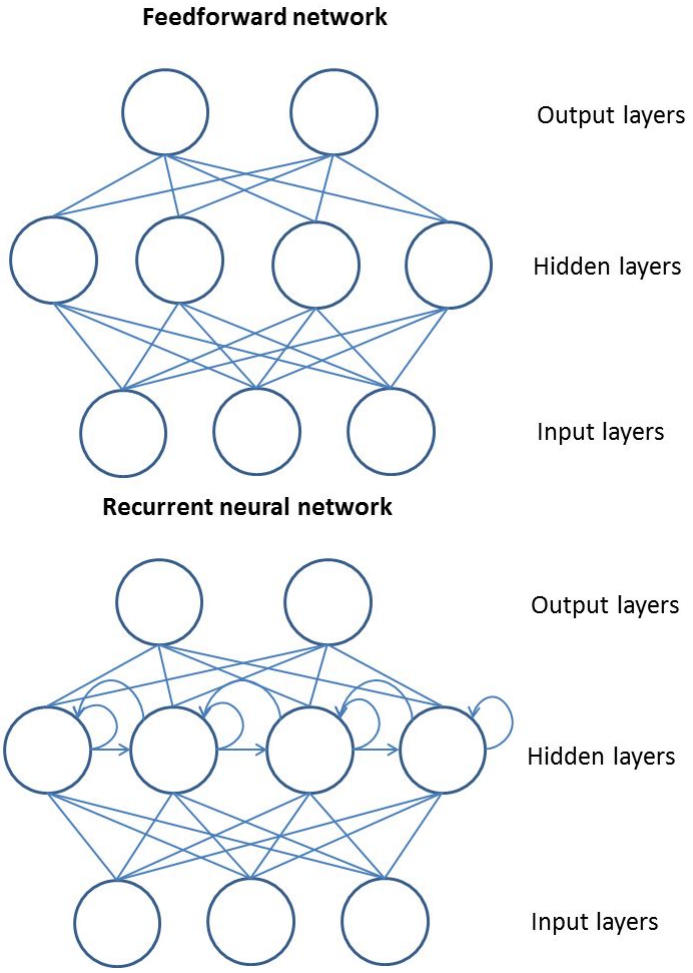


Figure 102.: Comparaison d'un feedforward network et d'un RNN

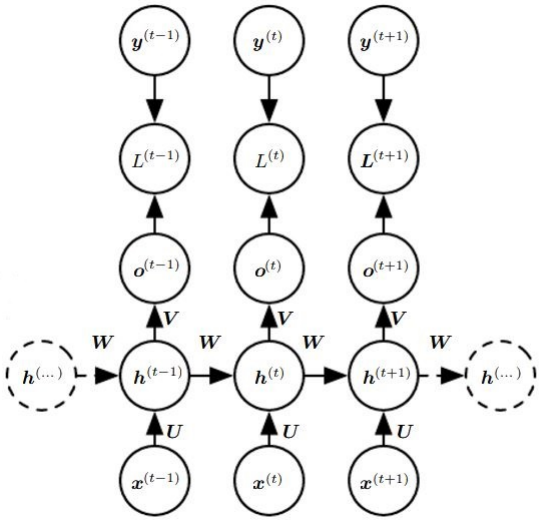


Figure 103.: Exemple de la manière d'entraîner un RNN. La fonction de perte L évalue la différence entre les sorties o du réseau de neurones et la séquence réelle y. Cette fonction de perte permet d'entraîner le réseau par back-propagation appelé *back-propagation through time*. [Goodfellow et al. 2016]

Plus formellement, le RNN illustré en figure 103 peut être caractérisé par les équations suivantes:

$$
\begin{aligned}
a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t)} \\
h^{(t)} &= tan(a^{(t)}) \\
o^{(t)} &= c + Vh^{(t)} \\
\hat{y}^{(t)} &= \text{softmax}(o^{(t)})
\end{aligned}
$$

avec b, c des constantes réelles W, U, V les matrices de paramètres des neurones considérés. La méthode d'optimisation d'un tel réseau est quasiment similaire à celle introduite au chapitre 2. Lorsqu'elle concerne les réseaux récurrents, la méthode d'optimisation peut également apparaître sous le terme anglais de *back-propagation through time* [Goodfellow et al. 2016].

Empiriquement, il a été constaté que la longueur des réseaux récurrents pouvait entraîner une atténuation brutale de l'information de sortie $h^{(t)}$. Ainsi certains réseaux récurrents tente d'y palier en formalisant notamment la notion de mémoire dans le filtre que constitue la fonction f. Ainsi, on distingue principalement dans les usages de réseaux récurrents les architectures de neurones suivantes : le *LSTM* (*Long short-term memory*) et le *GRU* (*Gated recurrent unit*).

Les deux réseaux ont des structures similaires bien que le GRU a la particularité d'être moins paramétré que le LSTM. Plus précisément, dans une couche du type LSTM, les neurones récurrents ont tous les mêmes paramètres. Par ailleurs, un neurone LSTM peut lui même être perçu comme la combinaison de plusieurs neurones. En effet un neurone LSTM comprend une combinaison de portes (on peut les considérer comme des neurones) qui jouant le rôle de filtre (cf Figure 104[1]).

1 `https://github.com/OKStateACM/AI_Workshop/wiki/LSTM-Generator`

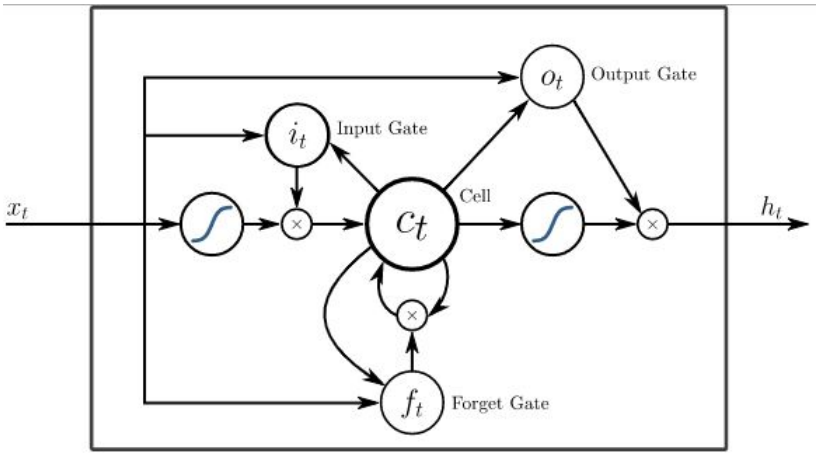


Figure 104.: Neurone LSTM, OKStateACM

Précisément, une neurone h_t d'une couche LSTM est définie par la séquence d'opérations suivante :

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

avec σ_g la fonction sigmoïde, σ_c et σ_h la fonction tangente hyperbolique, et $\circ$ l'opérateur d'Hadamard.

Les trois filtres f_t, i_t, o_t sont calculés comme l'image par la fonction sigmoïde d'une combinaison linéaire du signal passé h_{t-1} et de l'entrée x_t. Intuitivement, ces combinaisons s'inspire de l'électronique et permette d'atténuer un signal avant sa sortie (soit un équivalent d'oubli de h_{t-1} ou non).

D'autre part, le GRU peut formellement être définie comme la combinaison des opérations suivantes :

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)
\end{aligned}
$$

où les matrices W, U, b sont les paramètres du modèle.

Les RNN, LSTM (qui contiennent des couches LSTM) ou GRU peuvent de plus être bidirectionnels, c'est à dire prendre en compte le sens temporel croissant et décroissant (i.e. tenir compte de l'instant $t-1$ ou t). Cela permet ainsi d'introduire une dépendance dans une séquence aux valeurs adjacentes dans le temps.

A.4 QUELQUES PRÉCISIONS SUR LE CALIBRAGE DE NOTRE ALGORITHME DE DEEP LEARNING

A.4.0.1 *Format de l'image en entrée du réseau de neurones*

L'image fournie en entrée du réseau de neurones est de taille 128×64. Chaque image est transformée en noir et blanc c'est-à-dire que les élements d'entrée évoluent dans I_1. Les images de notre base de données ayant une taille supérieure sont re-dimensionnées, celles qui ont une taille inférieure sont remplies de 0 sur les bords. Dans la dimension de la largeur (64), il n'est pas obligatoire, d'avoir une image ayant une largeur exactement égale à 64. En effet, l'aspect séquentiel du réseau et la couche de neurones de sortie CTC permettent d'avoir des images de largeurs inférieurs à 64.

A.4.0.2 *Choix des méta-paramètres*

Dans ce paragraphe, nous exposons l'ensemble des paramètres qui ont pu être choisis dans le calibrage de notre algorithme. Nous utilisons ici un vocabulaire couramment utilisé dans les différentes librairies de *deep learning* comme celle de *Keras* que nous avons utilisée avec Python.

Deux méta-paramètres interviennent dans la structure du réseau : le nombre de convolutions qui est de 16 et le nombre de séquences de neurones GRU qui est de 512. Ce sont les paramètres qui nous ont donné les meilleurs résultats lors de nos différents essais.

D'autre part, afin d'entraîner correctement le réseau, certains méta-paramètres sont à choisir judicieusement. Le nombre d'images utilisées pour chaque itération de la *back-propagation* (*minibatch_size*) est de 32, ce nombre est en pratique déterminé empiriquement afin de mettre à jour l'ensemble des paramètres tout en limitant l'explosion du gradient. Le *learning_rate* choisi est de 0.01 et diminue de 10^{-6} au fur et à mesure de l'entraînement.

Le nombre d'itérations utilisées dans le cadre de notre optimisation des coefficients par descente de gradient stochastique (appelé *epoch_size* et introduit dans le chapitre 2) est choisi en utilisant l'*early_stopping*. Ainsi, la descente de gradient est arrêté lorsqu'une

décroissance des performances est observée.

Par ailleurs, l'ensemble des caractères que nous reconnaissons sont les caractères suivants (exprimé en expressions régulières) : [A-Za-z0-9] soit 62 classes auxquelles s'ajoutent les accents et les caractères spéciaux comme "/" ou "-". La dernière classe du réseau correspond à la classe vide.

A.4.0.3 *Utilisation de GPU pour l'entraînement du réseau*

De manière générale, un ordinateur est constitué d'un processeur (CPU) et d'une carte graphique (GPU), les calculs étant faits par le CPU et le traitement graphique étant fait par le GPU. Dans la cas des réseaux de neurones, il a été démontré que la capacité de calcul de GPU est supérieur à celle des CPU [Owens et al. 2008]. Un réseau de neurones peut ainsi être calibré de manière plus rapide.

L'entraînement sur GPU, dans notre cas, s'est fait sur une instance virtuelle[2] (ordinateurs virtuels) d'*Amazon Web Service* (AWS), préconfigurée à cet effet. Les packages python pour le *deep learning* tels que *keras* [3] ont ainsi pu être utilisés.

2 g2.2xlarge

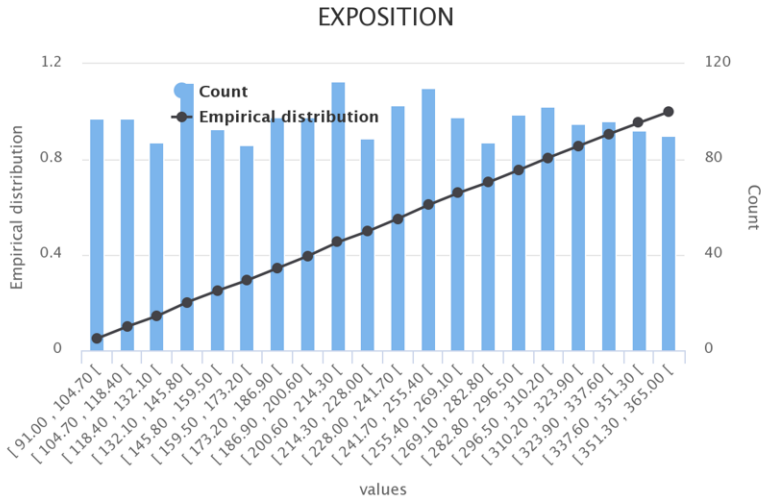
3 `https://keras.io/`

A.5 PRÉCISIONS SUR LES DONNÉES DU CHAPITRE 5

La nomenclature des données utilisées dans le chapitre 5 sont présentées ci-dessous ainsi que quelques analyses descriptives sur ces dernières.

- `POL_NUM` correspond au numéro de la police ($\sim$ ID).
- `CAL_YEAR` spécifie l'année de souscription du contrat.
- `EXPOSITION` correspond à l'exposition en jours de la police ($\in [\![91, 365]\!]$).
- `DRIVER_GENDER` spécifie le genre du conducteur/assuré (*Booléen*).
- `DRIVER_OCC` défini la statut professionnel de l'assuré ($\subseteq$ {'Employed', 'Unemployed', 'Self-employed', 'Housewife', 'Retired'}).
- `DRIVER_AGE` correspond à l'âge du conducteur/assuré (en années).
- `VEH_TYPE` défini le type de véhicule conduit (6 modalités allant de A à F).
- `VEH_CATEGORY` spécifie la catégorie du véhicule ($\subseteq$ {'Small', 'Medium', 'Large'}).
- `VEH_GROUP` correspond au groupe du véhicule ($\in [\![1, 20]\!]$)
- `VEH_VALUE` correspond à la valeur de véhicule ($\in [\![1000, 49995]\!]$).
- `POL_DURATION` défini l'ancienneté du contrat ($\in [\![0, 15]\!]$).
- `BONUS_MALUS` spécifie le montant de BONUS/MALUS du conducteur ($\in [\![-50, 150]\!]$).
- `DAMAGE_GUAR` correspond à l'indicateur d'une garantie dommage (*Booléen*).
- `STATE` défini la région de résidence du conducteur (10 modalités)
- `COUNTY` spécifie la sous-région de résidence du conducteur (471 modalités)
- `COUNTY_DENSITY` correspond à la densité de population de la sous-région de résidence du conducteur ($\in [14.38, 297.39]$)
- `NUM_LIAB_DAM` correspond aux nombres de sinistres RC matériels ($\in [\![0, 7]\!]$).
- `NUM_LIAB_BOD` correspond aux nombres de sinistres RC corporels ($\in [\![0, 3]\!]$).

- `INC_LIAB_DAM` correspond au coût total des sinistres RC matériels ($\in [0, 12878.4]$).
- `INC_LIAB_BOD` correspond au coût total des sinistres RC corporels ($\in [0, 69068]$).



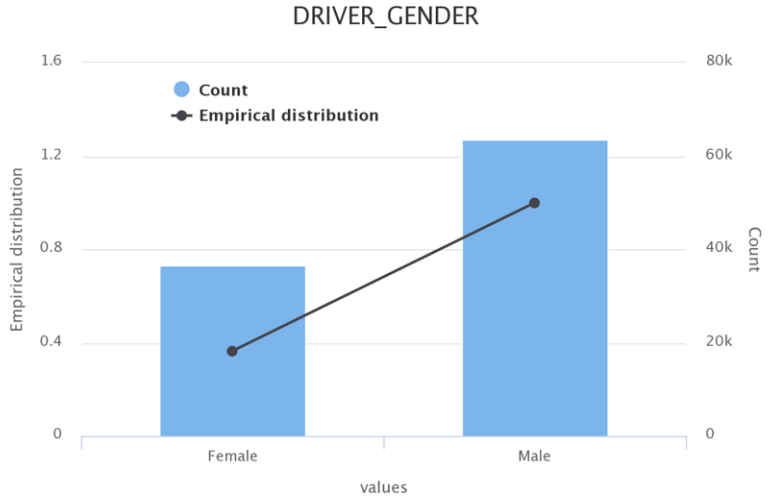
Quantiles

	quantiles	values
0	0.5%	96
1	10.0%	192
2	25.0%	340
3	50.0%	365
4	75.0%	365
5	90.0%	365
6	99.5%	365

Statistics

	statistics	values
0	min	91
1	max	365
2	mean	327.58025
3	stddev	73.57040591150603

Type : numeric
Number of modality : 275
Advise as numeric : true



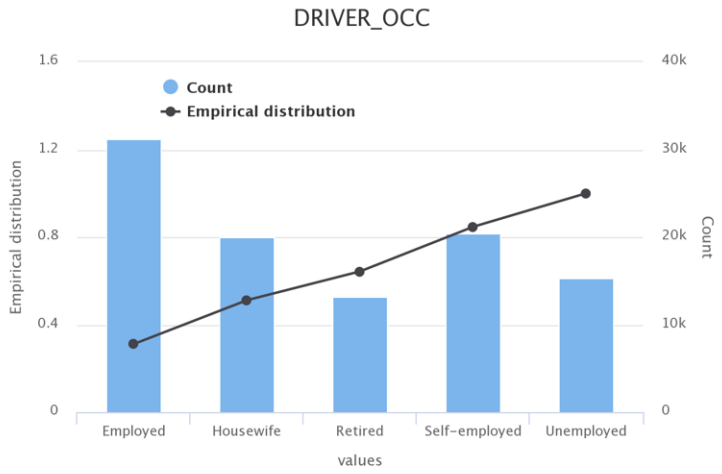
Less frequent

	Values	Count
0	Female	36568

Most frequent

	Values	Count
0	Male	63432

Type : string
Number of modality : 2
Advise as numeric : false



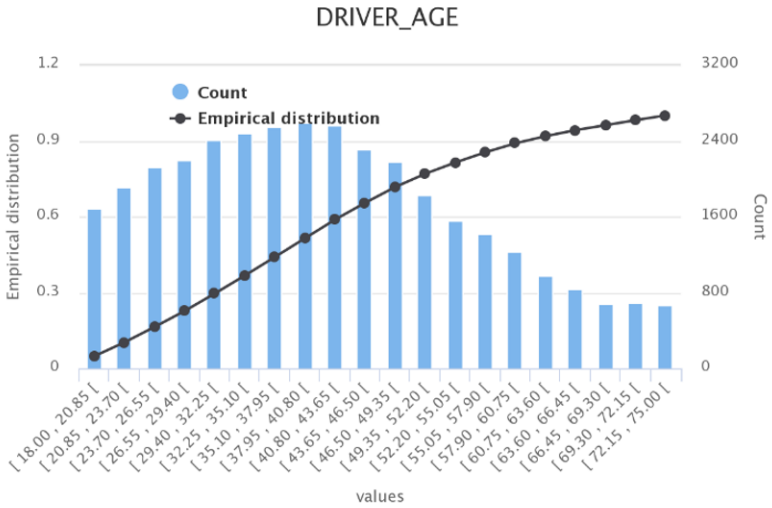
Less frequent

	Values	Count
0	Retired	13167
1	Unemployed	15315

Most frequent

	Values	Count
0	Housewife	20008
1	Self-employed	20371
2	Employed	31139

Type : string
Number of modality : 5
Advise as numeric : false



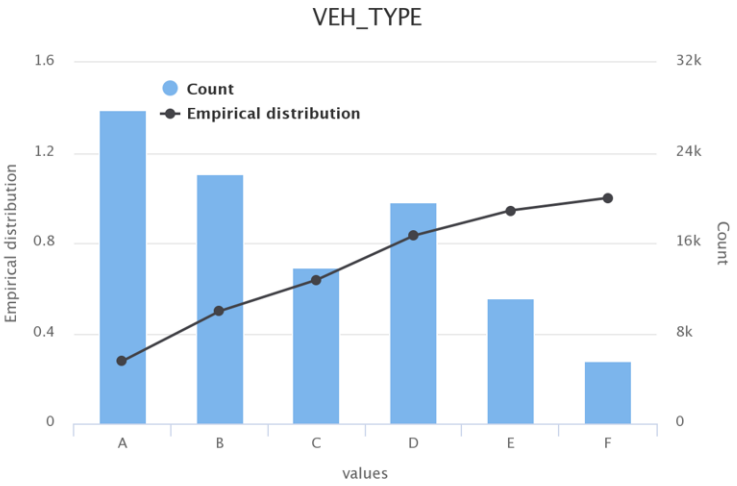
Quantiles

	quantiles	values
0	0.5%	18
1	10.0%	23
2	25.0%	30
3	50.0%	40
4	75.0%	51
5	90.0%	62
6	99.5%	75

Statistics

	statistics	values
0	min	18
1	max	75
2	mean	41.12506
3	stddev	14.298972851315092

Type : numeric
Number of modality : 58
Advise as numeric : true



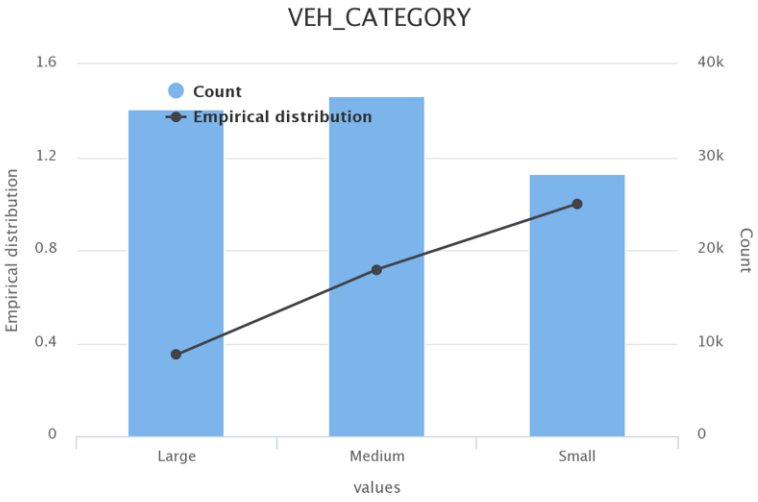
Less frequent

	Values	Count
0	F	5541
1	E	11168
2	C	13858

Most frequent

	Values	Count
0	D	19589
1	B	22088
2	A	27756

Type : string
Number of modality : 6
Advise as numeric : false



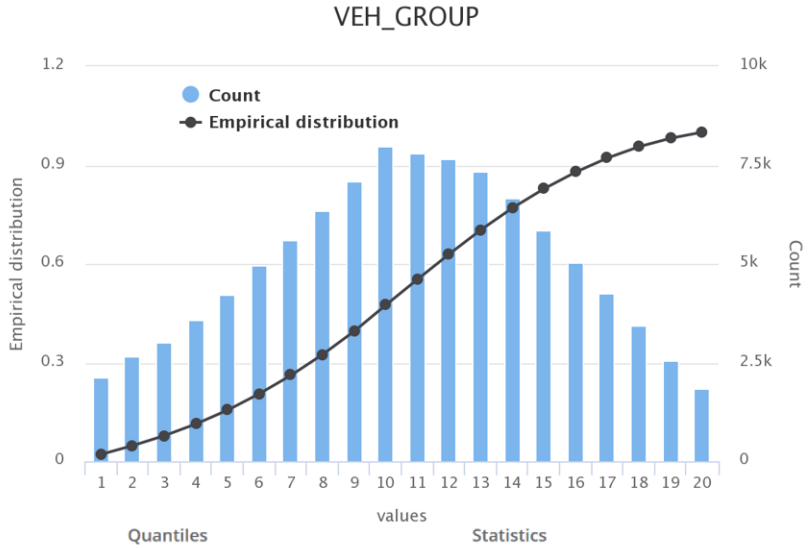
Less frequent

	Values	Count
0	Small	28238

Most frequent

	Values	Count
0	Large	35123
1	Medium	36639

Type : string
Number of modality : 3
Advise as numeric : false



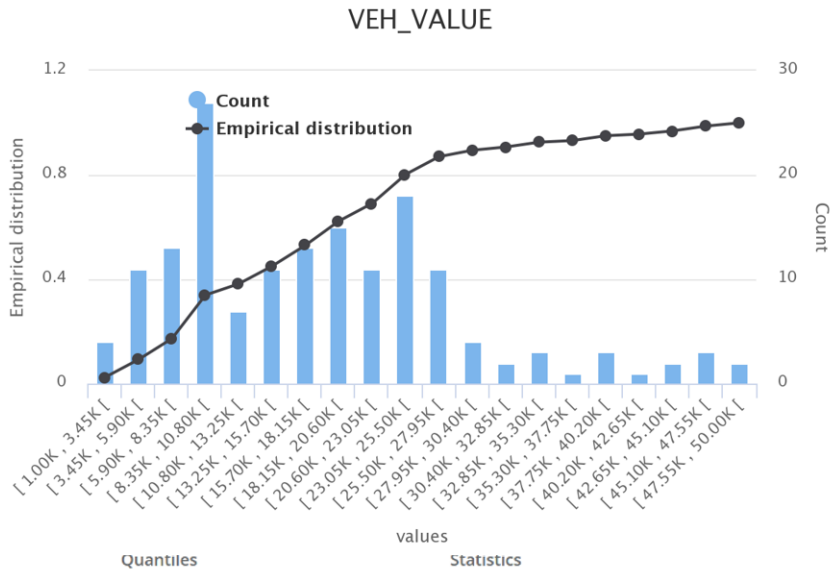
Quantiles

	quantiles	values
0	0.5%	1
1	10.0%	4
2	25.0%	7
3	50.0%	11
4	75.0%	14
5	90.0%	17
6	99.5%	20

Statistics

	statistics	values
0	min	1
1	max	20
2	mean	10.69284
3	stddev	4.687373726205804

Type : numeric
Number of modality : 20
Advise as numeric : true



Quantiles

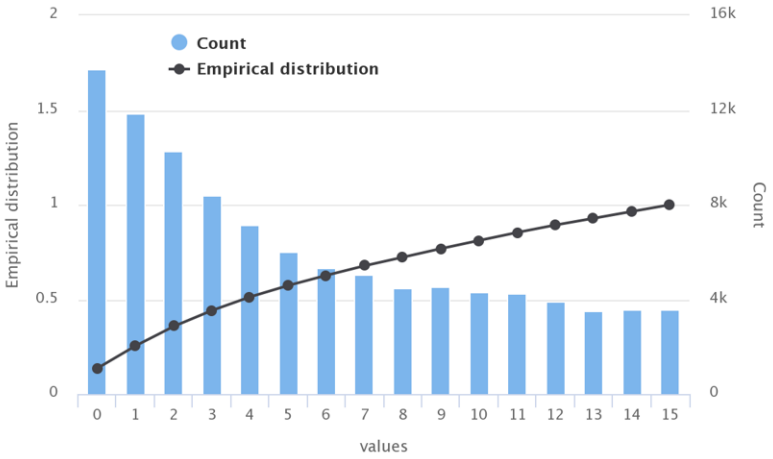
	quantiles	values
0	0.5%	1220
1	10.0%	5130
2	25.0%	8380
3	50.0%	14610
4	75.0%	22575
5	90.0%	29510
6	99.5%	48930

Statistics

	statistics	values
0	min	1000
1	max	49995
2	mean	16454.6194
3	stddev	10507.019806457769

Type : numeric
Number of modality : 939
Advise as numeric : true

POL_DURATION



Quantiles

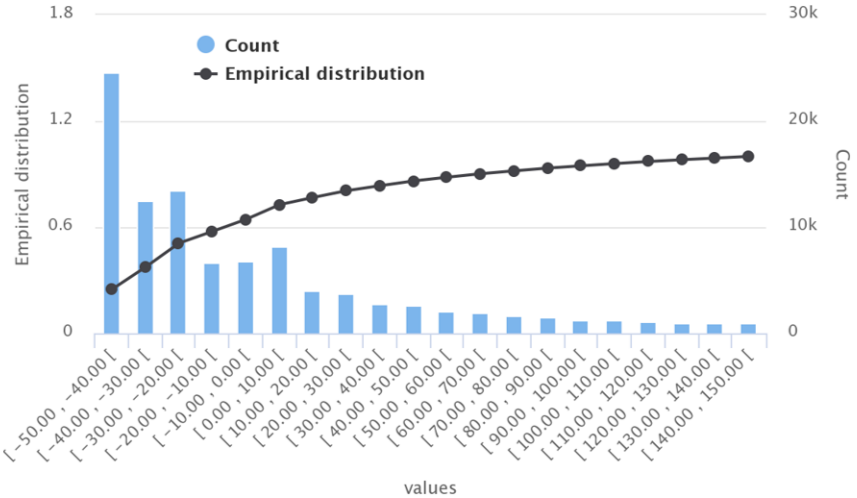
	quantiles	values
0	0.5%	0
1	10.0%	0
2	25.0%	1
3	50.0%	4
4	75.0%	9
5	90.0%	13
6	99.5%	15

Statistics

	statistics	values
0	min	0
1	max	15
2	mean	5.47093
3	stddev	4.591155162065121

Type : numeric
Number of modality : 16
Advise as numeric : true

BONUS_MALUS



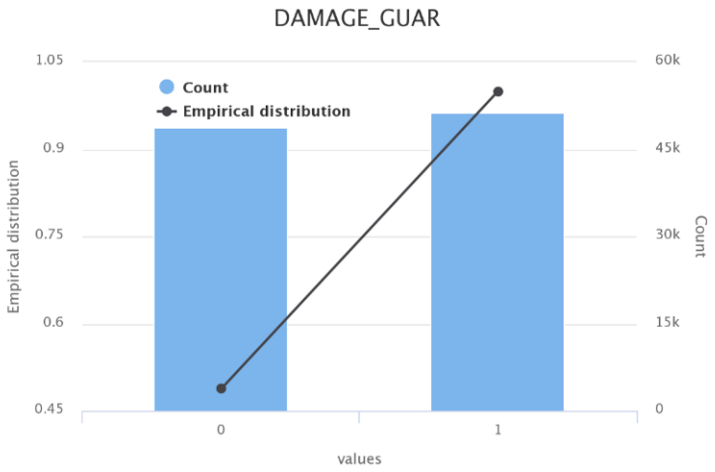
Quantiles

	quantiles	values
0	0.5%	-50
1	10.0%	-50
2	25.0%	-40
3	50.0%	-30
4	75.0%	10
5	90.0%	70
6	99.5%	150

Statistics

	statistics	values
0	min	-50
1	max	150
2	mean	-6.93
3	stddev	48.62794203714018

Type : numeric
Number of modality : 21
Advise as numeric : true



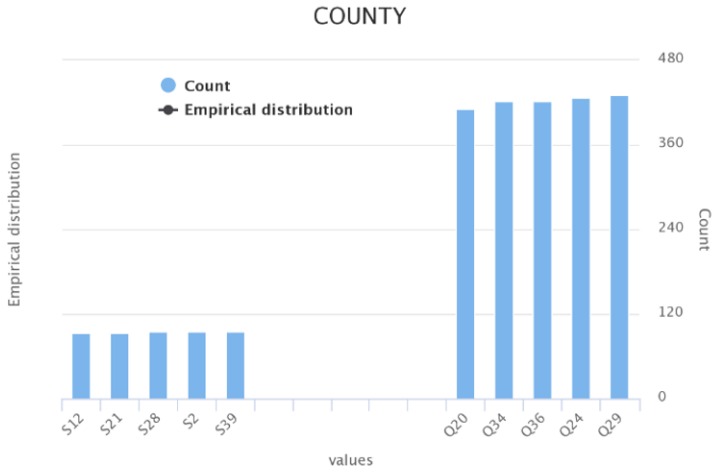
Quantiles

	quantiles	values
0	0.5%	0
1	10.0%	0
2	25.0%	0
3	50.0%	1
4	75.0%	1
5	90.0%	1
6	99.5%	1

Statistics

	statistics	values
0	min	0
1	max	1
2	mean	0.5122
3	stddev	0.4998536371144911 6

Type : numeric
Number of modality : 2
Advise as numeric : true



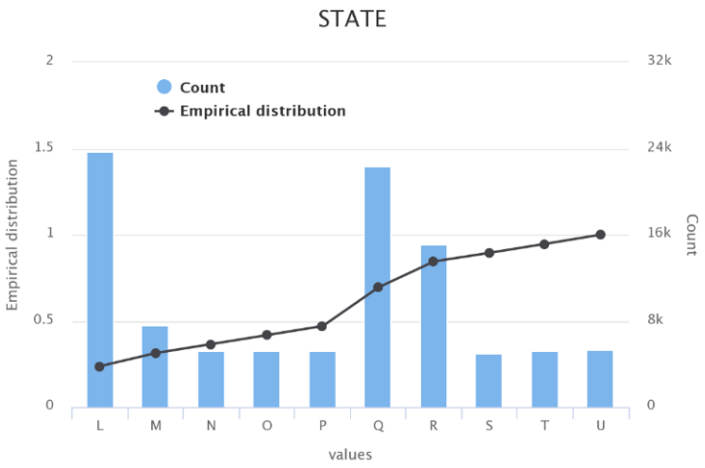
Less frequent

	Values	Count
0	S12	94
1	S21	94
2	S28	95
3	S2	96
4	S39	96

Most frequent

	Values	Count
0	Q20	412
1	Q34	421
2	Q36	422
3	Q24	427
4	Q29	431

Type : string
Number of modality : 471
Advise as numeric : false



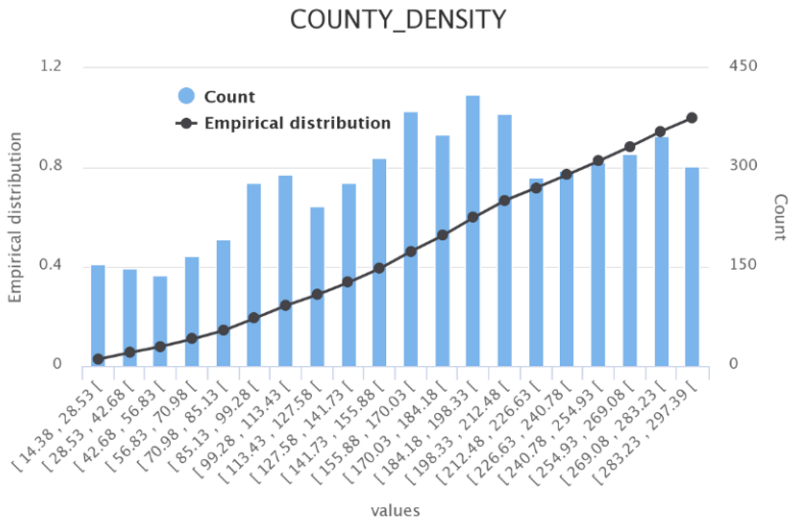
Less frequent

	Values	Count
0	S	4994
1	N	5193
2	T	5195
3	O	5213
4	P	5259

Most frequent

	Values	Count
0	U	5364
1	M	7595
2	R	15076
3	Q	22381
4	L	23730

Type : string
Number of modality : 10
Advise as numeric : false



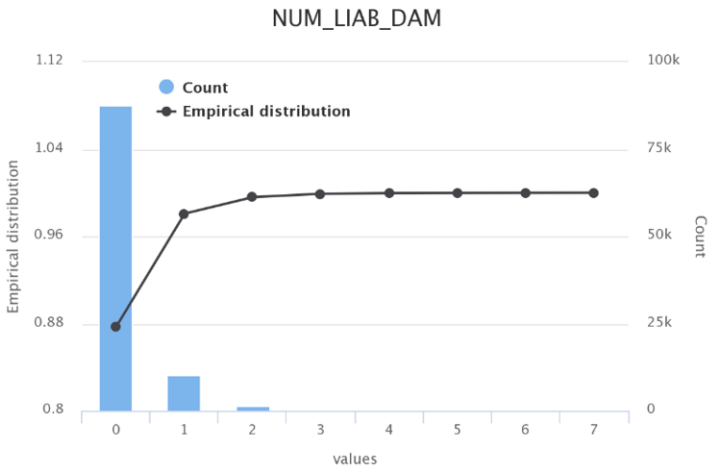
Quantiles

	quantiles	values
0	0.5%	17.8799582
1	10.0%	30.18943347
2	25.0%	50.6257826
3	50.0%	94.36462339
4	75.0%	174.6445246
5	90.0%	240.0004036
6	99.5%	296.4319078

Statistics

	statistics	values
0	min	14.37714238
1	max	297.3851697
2	mean	117.15688049021138
3	stddev	79.49898804465192

Type : numeric
Number of modality : 471
Advise as numeric : true



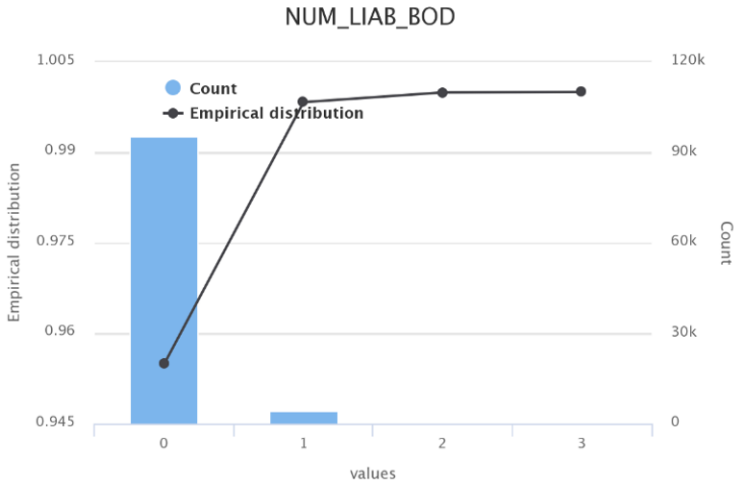
Quantiles

	quantiles	values
0	0.5%	0
1	10.0%	0
2	25.0%	0
3	50.0%	0
4	75.0%	0
5	90.0%	1
6	99.5%	2

Statistics

	statistics	values
0	min	0
1	max	7
2	mean	0.14724
3	stddev	0.4366947325339457

Type : numeric
Number of modality : 8
Advise as numeric : true



Quantiles

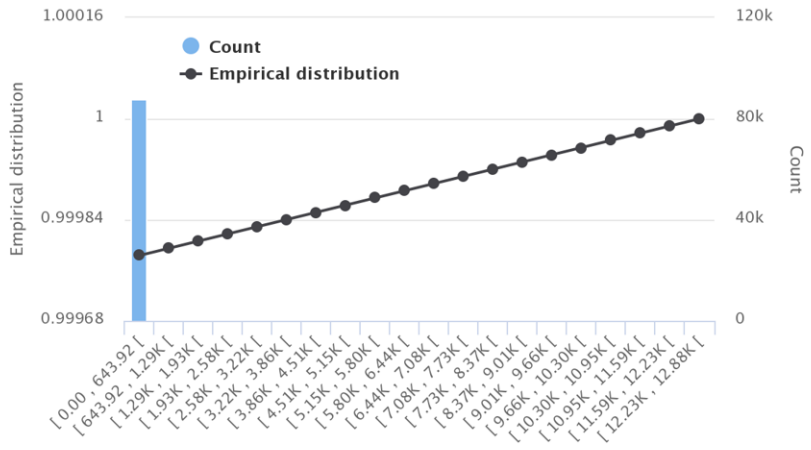
	quantiles	values
0	0.5%	0
1	10.0%	0
2	25.0%	0
3	50.0%	0
4	75.0%	0
5	90.0%	0
6	99.5%	1

Statistics

	statistics	values
0	min	0
1	max	3
2	mean	0.04678
3	stddev	0.21952702230280272

Type : numeric
Number of modality : 4
Advise as numeric : true

INC_LIAB_DAM



Quantiles

	quantiles	values
0	0.5%	0
1	10.0%	0
2	25.0%	0
3	50.0%	0
4	75.0%	0
5	90.0%	160.3307466
6	99.5%	2911.205121

Statistics

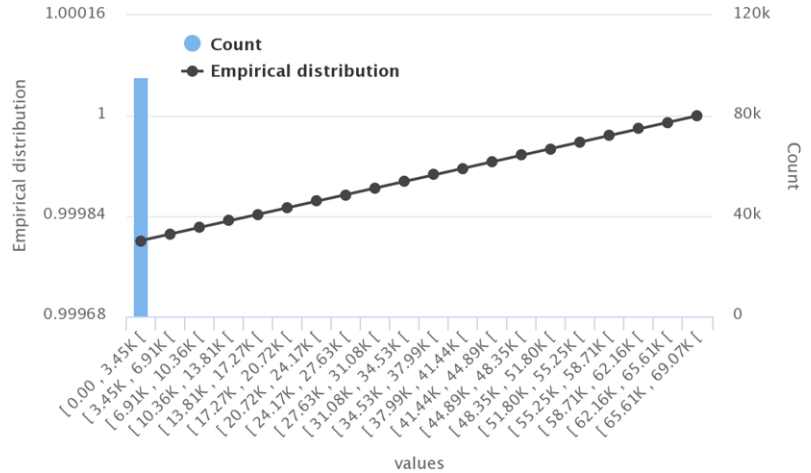
	statistics	values
0	min	0
1	max	12878.36991
2	mean	106.15729505072343
3	stddev	444.9932471326943

Type : numeric

Number of modality : 12257

Advise as numeric : true

INC_LIAB_BOD



Quantiles

	quantiles	values
0	0.5%	0
1	10.0%	0
2	25.0%	0
3	50.0%	0
4	75.0%	0
5	90.0%	0
6	99.5%	12447.40846

Statistics

	statistics	values
0	min	0
1	max	69068.02629
2	mean	222.74327196709487
3	stddev	1859.5625777467199

Type : numeric

Number of modality : 4505

Advise as numeric : true

A.6 LES MODÈLES LINÉAIRES GÉNÉRALISÉS (GLM)

Cette section s'inspire du cours de l'ENSAE d'Olivier Lopez.

Définition A.1. Un modèle statistique $(\Omega, \mathcal{F}, (\mathbb{P}_{\theta,\phi})_{\theta\in\Theta,\phi>0})$ est dit exponentiel si la mesure de probabilité $\mathbb{P}_{\theta,\phi}$ admet une densité $f_{\theta,\phi}$ selon une mesure dominante μ (le plus souvent la mesure de Lebesgue) tel que :

$$f_{\theta,\phi}(y) = c_\phi(y) \exp\left(\frac{y\theta - a(\theta)}{\phi}\right)$$

θ est appelé le paramètre canonique et ϕ la paramètre de dispersion (souvent considéré comme un paramètre de nuisance). On note également que $a(\theta)$ est $\mathcal{C}^2$ et convexe et que $c_\phi(y)$ ne dépend pas de θ.

Dans le cadre de modèles linéaires généralisés, en posant la variable à expliquer $Y \in \mathbb{R}$ et le vecteur de covariables $X \in \mathbb{R}^d$, réaliser une régression linéaire généralisée de Y à partir de X suppose deux hypothèses :

1. $Y|X = x \sim \mathbb{P}_{\theta(x),\phi}$ où $\mathbb{P}_{\theta(x),\phi}$ appartient à une famille exponentielle.

2. $g(\mathbb{E}[Y|X]) = \beta^T X$ avec g une fonction bijective.

On appelle g la fonction de lien associée à la distribution de $Y|X$, et alors que nous avons comme propriété des familles exponentielles que $\mathbb{E}[Y|X] = a'(\theta)$, on peut réecrire le point 2. tel que $g(a'(\theta(X))) = \beta^T X$. On en déduit alors une fonction de lien canonique pour chaque loi issue d'une famille exponentielle en posant $g(t) = a'^{-1}(t)$, on obtient alors $\theta(X) = \beta^T X$. Les principales distributions issues de familles exponentielles ainsi que leurs fonctions de liens canoniques sont répertoriés dans le tableau 24.

Table 24.: Principales distributions exponentielles et fonctions canoniques associées

Distribution	$g(\mu)$	Lien canonique
Gaussienne	μ	Indentité
Poisson	$\ln(\mu)$	Log
Gamma	μ^{-1}	Inverse
Binomial	$\log(\mu/(1-\mu))$	Logit

A.7 PRÉCISION SUR LA CONSTRUCTION DU MODÈLE COÛT/FRÉQUENCE

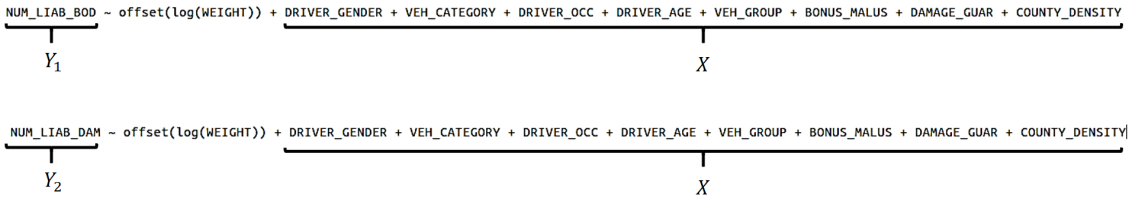
Cette section détaille la paramétrisation des GLM utilisés dans la modélisation de la prime pure.

Les fréquences des sinistres corporels et matériels sont modélisés avec une distribution binomiale négative avec la fonction log comme lien. Alors que nous avions commencé par utiliser un GLM poisson, nous nous sommes rendu compte que les résultats des prédictions (MSE) étaient légèrement meilleurs en employant une distribution binomiale. A noter que le choix d'une distibution binomiale négative permet notamment de modéliser des effets de surdispersion des données (lorsque l'on a $\mathbb{E}[Y|X] < \text{Var}(Y|X)$).

Les fréquences observées sont corrigées de l'exposition. Ainsi nos variables Y_1 et Y_2 à prédire (soit respectivement `NUM_LIAB_DAM` et `NUM_LIAB_BOD`) sont divisées par la variable `WEIGHT`. Ainsi un individu qui n'a été assuré au sein de la compagnie que la moitié de l'année (`EXPOSITION`=183, soit `WEIGHT` $\sim$ `0.5`) et qui compte deux sinistres au cours de cette période verra sa sinistralité normalisée sur l'année s'élever à 4. Le support de la loi binonmiale étant $\mathbb{N}$, on introduit alors un paramètre `Offset` dans nos modèles lors du calibrage afin de corriger les variables non entières introduites par la prise en compte de l'exposition. En effet, la fonction de lien utilisée étant $g = \ln()$, en posant X le vecteur des variables explicatives du modèle, on peut donc écrire:

$$g\left(\frac{\mathbb{E}[Y|X]}{\texttt{WEIGHT}}\right) = \beta^T X \Leftrightarrow \ln(\mathbb{E}[Y|X]) = \underbrace{\ln(\texttt{WEIGHT})}_{\texttt{Offset}} + \beta^T X \tag{40}$$

Ainsi la modélisation des fréquences de sinistres corporels et matériels devient :



Les variables quantitatives continues `DRIVER_AGE` et `COUNTY_DENSITY` notamment, n'ont pas été discrétisées comme il est souvent d'usage en tarification afin de faciliter l'élaboration du modèle.

Les coûts moyens des sinistres basés sur les variables `MEAN_INC_BOD` et `MEAN_INC_DAM` sont retraités d'un écrêtemment afin "de redistribuer" les montants extrêmes. Pour ce faire nous avons donc affecté à tous les sinistres corporels supérieurs à 6861,94€ (valeur du quantile 99%, soit les polices dont les sinistres corporels sont parmi les 1% les plus élevés) ce montant maximal. Cette méthode d'écrêtage est employée ici pour sa simplicité et est parfois d'usage opérationnel dans le milieu actuariel. Cette heuristique permet de

faciliter le calibrage des paramètres dans le cas d'observation de valeurs extrêmes. Par la suite on calcul un facteur de correction tel que :

$$\text{Correction} = \frac{\sum_{i=1}^{100000} \texttt{INC\_IAB\_BOD}_i - \sum_{i=1}^{100000} \texttt{MEAN\_INC\_BOD}_i.\texttt{NUM\_LIAB\_BOD}_i}{\sum_{i=1}^{100000} \texttt{WEIGHT}_i}$$

Que l'on appliquera en amont de la prédiction tel que :

$$(\hat{Y_4}.\hat{Y_2})_{\text{finale}} = \hat{Y_4}.\hat{Y_2} + \text{Correction}$$

On fait de même pour les coûts moyens matériels, avec un seuil de 2278,65€ (quantile à 99,5%).

Finalement, pour prédire les variables Y_3 et Y_4, on utilise un GLM avec une distribution inverse-gaussienne et une fonction de lien *inverse*. En ne considérant que les variables précédentes filtrées pour ne conserver que les polices d'assurances sinistrées (dont les montants sont strictement positifs) on obtient la modélisation des coûts moyens des sinistres corporels et matériels suivant :

$$\underbrace{\texttt{MEAN\_INC\_DAM}}_{Y_3} \sim \underbrace{\texttt{DRIVER\_GENDER} + \texttt{VEH\_TYPE} + \texttt{VEH\_CATEGORY} + \texttt{DRIVER\_OCC} + \texttt{DRIVER\_AGE}}_{X}$$

$$\underbrace{\texttt{MEAN\_INC\_BOD}}_{Y_4} \sim \underbrace{\texttt{DRIVER\_GENDER} + \texttt{VEH\_TYPE} + \texttt{VEH\_CATEGORY} + \texttt{DRIVER\_OCC} + \texttt{DRIVER\_AGE}}_{X}$$

Ainsi afin d'obtenir la charge sinistre totale prédite pour chaque police au sein de notre modèle il nous suffit alors de calculer :

$$\text{Charge sinistre}_{\text{Pred}} = (\hat{Y_1}.\hat{Y_3})_{\text{finale}} + (\hat{Y_4}.\hat{Y_2})_{\text{finale}}$$

A.8 COMPLÉMENTS SUR LES MÉTHODES D'APPRENTISSAGE NON-SUPERVISÉES

A.8.1 *Résultats des partitions obtenues à l'aide des K-means*

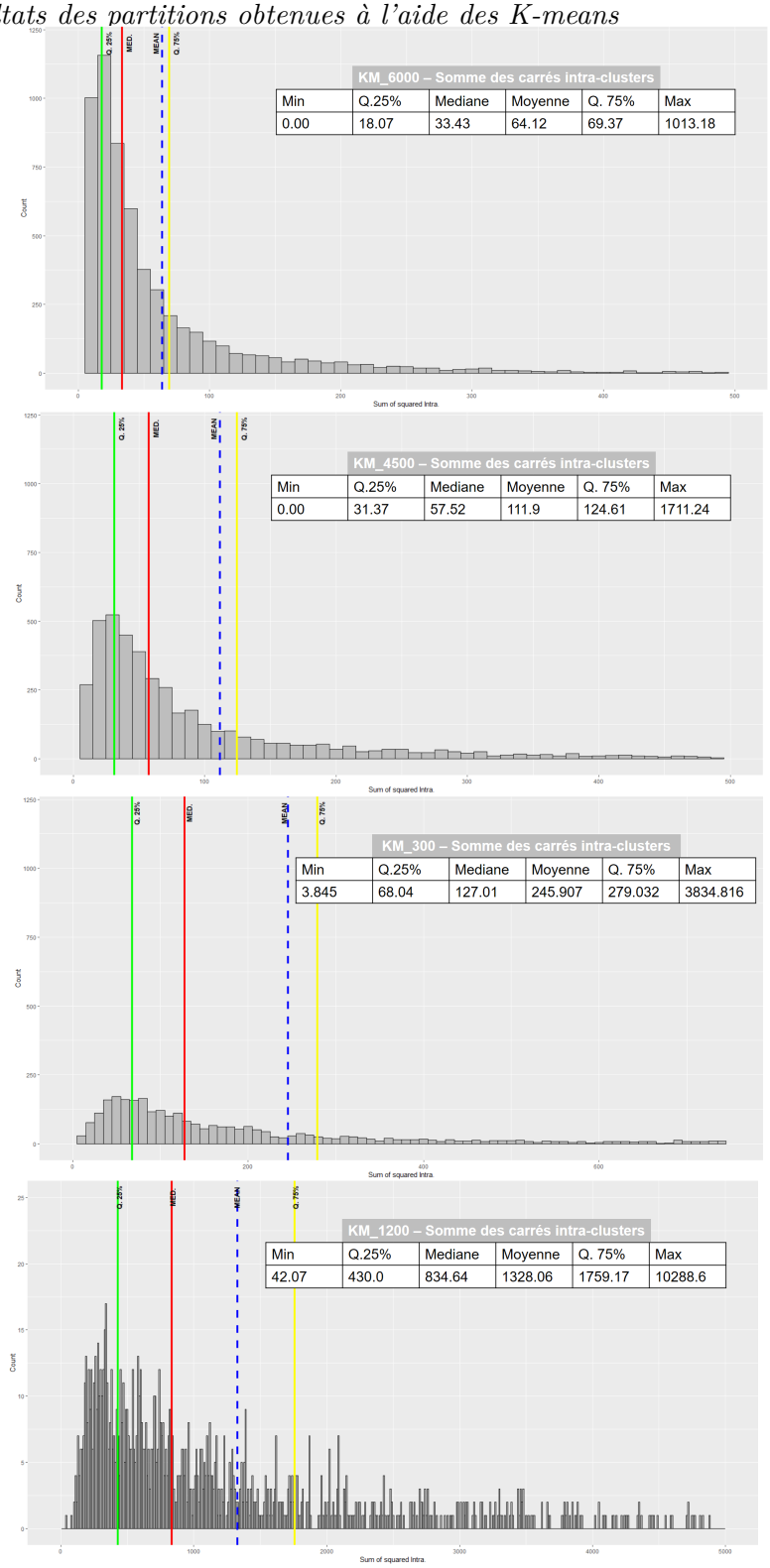


Figure 105.: Répartition de la somme des carrés intra-clusters pour les différents partitionnements par l'algorithme du K-means, où $k \in \{6000, 4500, 3000, 1200\}$

A.8.2 *La Classification à Ascendance Hiérarchique*

L'approche du clustering hiérarchique est fondée sur une mesure de dissimilarité entre les observations, notée par la suite $d_{x_i x_j}$, qui peut par exemple être la distance usuelle. Il existe deux types d'algorithmes de clustering hiérarchique :

- La *méthode ascendante* : au départ toutes les observations consituent un groupe. Ces classes sont alors fusionnées deux à deux en fonction de leur similarité. La récurrence se poursuit jusqu'à l'obtention du groupe trivial (i.e. toute la donnée). L'algorithme conserve alors en mémoire les différents états du partitionnement à chaque récurrence permettant à l'utilisateur d'arbitrer la classification qui lui correspond le mieux. Nous utilisons cette méthode dans les résultats présentés dans le chapitre 5.
- La *méthode descendante* : par déduction de l'approche précédente, l'algorithme commence alors avec le groupe trivial et scinde de manière récurrente en deux sous-groupes les plus distants possibles jusqu'à obtenir autant de classes que d'individus.

La mesure de dissimilarité entre deux groupes s'appuie généralement sur une distance d et peut se définir de différentes manières. Plusieurs distances ont ainsi été testées. Si pour tout entier k, C_k représente un sous groupe formé lors de classification à ascendance hiérarchique, on définit alors les distances inter-classes suivantes :

$$d_{C_k C_l} = \min_{x_i \in C_k, x_j \in C_l} d_{x_i x_j}$$

$$d_{C_k C_l} = \max_{x_i \in C_k, x_j \in C_l} d_{x_i x_j}$$

$$d_{C_k C_l} = \frac{1}{|n_k|} \frac{1}{|n_l|} \sum_{x_i \in C_k} \sum_{x_j \in C_l} d_{x_i x_j}$$

$$d_{C_k C_l} = d_{\mu_k \mu_l}$$

où pour tout entier k, μ_k représente le barycentre du sous-ensemble C_k. Ces distances nous ont fournis des résultats peu probants en terme de classification. Suivant nos critères définis dans le chapitre 5, la classification la plus pertinente nous a amené à considérer 2320 clusters comptant tous plus d'une observation. L'homogénéité du partitionnement que nous appellons HCLUST_2320 est réprésentée ci-dessous en figure 106.

A.8.3 *Résultats de partitionnement obenus avec OPTICS*

Nous avons alors réalisé deux partitionnements avec MinPoints $= 2$ et $\epsilon = 120$ puis où MinPoints $= 3$ et $\epsilon = 80$. Dans les deux cas nous avons choisi un seuil d'écrêtage de 2.

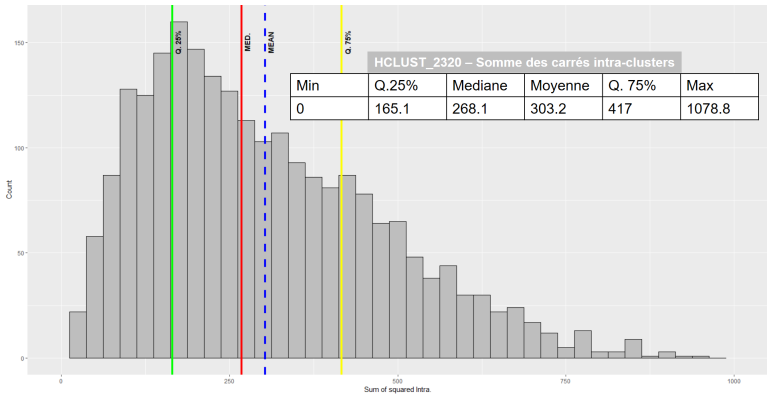


Figure 106.: Répartition de la somme des carrés intra-clusters (HCLUST_2320)

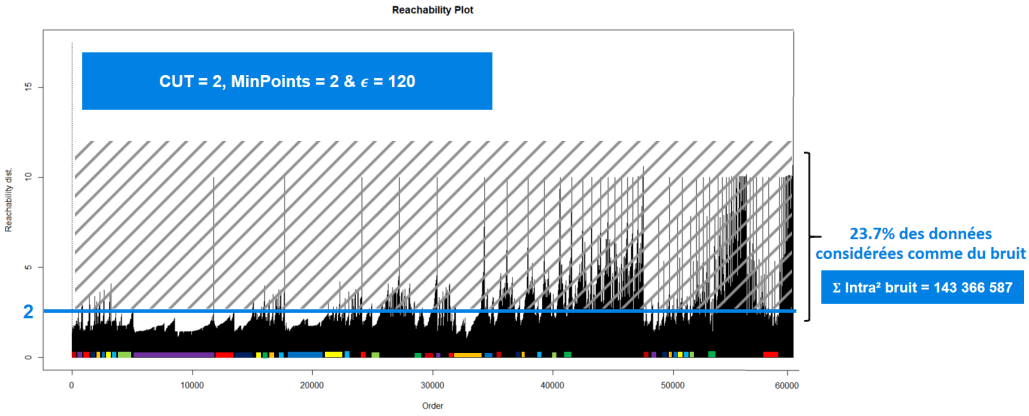
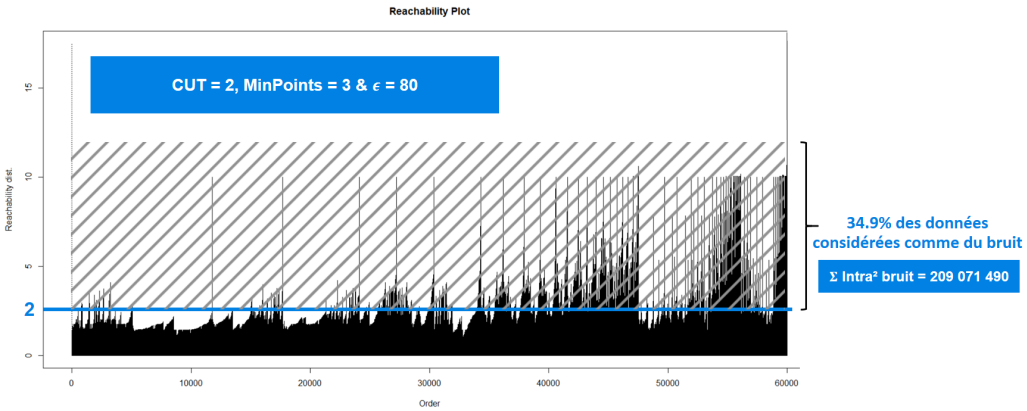


Figure 107.: Reachability-plot et résultats d'OPTICS(2,2,120).



OPTICS(2,3) – Somme des carrés intra-clusters *(hors bruit)*

Min	Q.25%	Mediane	Moyenne	Q. 75%	Max
0	1.6	4.3	1148.7	14.1	775912.7

Figure 108.: Reachability-plot et résultats d'OPTICS(2,3,80).

Les annexes suivantes s'inspirent des travaux de D. Delcaillau.

A.9 L'ALGORITHME XTREME GRADIENT BOOSTING

A.9.1 *Le Boosting*

Pour comprendre l'algorithme XGboost, il est nécessaire de décrire le Boosting [Freund et al. 1996] et notamment Adaboost introduit par Freund and Schapire (1997), ainsi que l'algorithme de Gradient Boosting [Friedman 2001]. Le Boosting est une technique ensembliste qui consiste à agréger des classifieurs dits faibles (dû à leur grande erreur) construits itérativement par pondération des observations en fonction des erreurs qu'elles engendrent sur la fonction de coût. L'idée est alors, en agrégeant différant classifieurs faibles, d'obtenir un classifeur avec un meilleur pouvoir de prédiction.

Ainsi, nous descrivons dans le paragraphe suivant, l'algorithme de Boosting (Adaboost) introduit par Freund and Schapire (1997) et expliqué dans [Schapire 2013]:

Soit B le nombre de modèles utilisés, (ALGO) l'algorithme utilisé (par exemples un arbre de décision), y la variable binaire cible $y \in -1, 1$, X la matrice des variables explicatives. On considère l'ensemble d'apprentissage : $(x_1, y_1), ..., (x_n, y_n)$, c'est-à-dire: $x = (x_{i,j})_{1 \leq i \leq n, 1 \leq j \leq p}$ et $(y_i)_{1 \leq i \leq n}$

- LISTE, une liste contenant les différents modèles initialement vide.
- les individus sont pondérés uniformément: $w_i^1 = 1/n$
- pour b = 1 à N:
  - construire le modèle M_b basé sur les données pondérées par w^b à l'aide de (ALGO).
  - ajouter M_b dans LISTE
  - calcul du taux d'erreur commis par M_b: $\epsilon_b = \sum_{i=1}^{n} w_i \mathbb{I}_{y_i \neq \hat{y_i}}$
  - si $\epsilon_b > 0.5$ ou $\epsilon_b = 0$: arrêt de l'algorithme
  - sinon:
    - calcul du poids du classifieur: $\alpha_b = ln(\frac{1-\epsilon_b}{\epsilon_b})$

$*$ calcul du poids suivant: $w_i^{b+1} = w_i^b e^{\alpha \mathbb{I}_{y_i \neq \hat{y}_i}}$

A la fin de cet algorithme, un vote pondéré par $(\alpha_b)_{1 \leq b \leq B}$ est effectué et finalement:

$$f(x) = sign(\sum_{b=1}^{B} \alpha_b M_b(x)) \tag{41}$$

où x est le vecteur des variables explicatives d'entrée, et $f(x)$ la sortie estimée par notre modèle de Boosting. Il existe de nombreux variantes sur le choix des des poids α des classifieurs.

Nous constatons ici que le choix sur du modèle utilisé (ALGO) est laissé libre. Schapire (2013) ne fournit alors qu'une notion de *weak learner* sans en donner une définition mathématique précise. En général, ce sont des arbres de décision qui sont utilisés car on peut moduler leurs propriétés, comme la profondeur de l'arbre par exemple et ainsi s'assurer du caractère explicatif faible du modèle.

A.9.1.1 *Algorithme d'optimisation de descente de gradient*

Pour faire le lien avec l'algorithme de gradient boosting, il est également nécessaire de rappeler l'algorithme de descente de gradient. Ce dernier introduit dans le chapitre 2 est une technique itérative qui permet d'approcher la solution d'un problème d'optimisation convexe.

Considérons une fonction différentiable f, d'espace de départ $\mathbb{R}^p$ (muni de la norme $\| \ . \ \|$ et du produit scalaire $< \ . \ >$) et d'espace d'arrivée $\mathbb{R}$, qu'on souhaite minimiser. Pour $x \in \mathbb{E}$, on note $\nabla f(x)$ le gradient de f en x: $\nabla f(x) = (\frac{\partial f}{\partial x_i}(x))_{1 \leq i \leq p}$. On choisit une valeur initiale $x_0 \in \mathbb{R}^p$ pour l'algorithme, ainsi qu'un seuil (critère d'arrêt) $\epsilon > 0$. L'algorithme va alors calculer de manière itérative des valeurs $x_1, x_2, ...$ jusqu'à convergence. Autrement dit, les itérations se décrivent selon la logique suivante :

- k=0
- tant que $\| \nabla f(x_k) \| > \epsilon$: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, avec α_k un paramètre de lissage.

Notons j une fonction de coût individuelle utilisée pour comparer la valeur $f(x_i) = \hat{y}_i$ prédite par notre modèle et la valeur y_i réelle (par exemple $j(y_i, f(x_i)) = (y_i - f(x_i))^2$ couramment utilisée en régression). Notons J la fonction de coût global, définie comme la somme des fonctions de coût individuelles: $J(y, f) = \sum_{i=1}^{n} j(y_i, f(x_i)$

Ainsi, pour reprendre l'algorithme d'Adaboost, si on note f_b notre classifieur à l'étape b, la descente de gradient peut se retranscrire aux travers de la formule d'optimisation suivante :

$$f_b(x) = f_{b-1}(x) - \alpha \nabla j(y, f(x)) \tag{42}$$

$$= f_{b-1}(x_i) - \alpha \frac{\partial j(y_i, f(x_i))}{\partial f(x_i)}, 1 \leq i \leq n \tag{43}$$

A.9.1.2 *Algorithme de Gradient Boosting*

CAS DE LA RÉGRESSION

Nous pouvons à présent définir la méthode de Gradient Boosting introduite par Friedman (2001), qui généralise l'approche d'optimisation en ligne introduite dans Adaboost. Les notations précédentes sont conservées, à savoir f_i pour les fonctions associées au modèles de l'itération i et J la fonction de coût à optimiser. L'algorithme se définit ainsi:

- initialisation par un modèle constant: $f_0(x) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} J(y_i, \theta)$ (correspondant à la moyenne des observations de y pour une fonction de coût correspondant à la norme euclidienne)
- pour b allant de 1 à B (on pourrait aussi introduire un critère d'arrêt):
  - calcul des pseudo résidus: $r_{i,m} = -\frac{\partial J(y_i, f(x_i))}{\partial f(x_i)}|_{f(x)=f_{m-1}(x)}, 1 \leq i \leq n$
  - ajustement d'un classifieur $h_m(x)$ (faible, comme un arbre par exemple), entrainé sur les données $(x_i, r_{i,m})_{1 \leq i \leq n}$
  - calcul du prochain classifieur: $f_m(x) = f_{m-1}(x) + \theta_m h_m(x)$, où θ_m est obtenu en résolvant le problème d'optimisation: $\theta_m = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} J(y_i, f_{m-1})(x) + \theta h_m(x_i)$

A la sortie de cet algorithme, on obtient: $f_M(x)$, notre classifieur final. Les fonctions de coût généralement choisies pour la régression sont:

- *écart quadratique*: $j(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$. Cette fonction présente l'avantage d'être sensible aux écarts mais l'inconvénient d'être peu robuste vis-à-vis des points abérrants
- *valeur absolue*: $j(y_i, f(x_i)) = |y_i - f(x_i)|$. L'avantage de ce choix réside dans sa robustesse aux points abérrants mais est moins sensible aux écarts
- *fonction Huber*: $j(y_i, f(x_i)) = \begin{cases} y_i - f(x_i) \ si \ |y_i - f(x_i)| < \delta_q \\ \delta_q sign(y_i - f(x_i)) \ sinon \end{cases}$ où δ_q est le quantile d'ordre q de $(|y_i - f(x_i)|)_{1 \leq i \leq n}$. Ceci permet d'obtenir les avantages de la valeur absolue (sensibilité aux valeurs faibles du gradient) et de l'écart quadratique (sensibilité aux valeurs élevées du gradient).

CAS DE VARIABLE CIBLE CATÉGORIELLE

Dans le cas où la variable cible est catégorielle avec K modalités (de 1 à K), l'algorithme reste très proche de celui présenté précédemment, avec néanmoins une fonction de coût différente, adaptée au classement. Pour cela on définit: $y_i^k = \begin{cases} 1 \ si \ y_i = k \\ 0 \ sinon \end{cases}$, $1 \leq i \leq n, 1 \leq k \leq K$. La nouvelle fonction de coût est alors définie par:

$$j(y_i, f(x_i)) = -\sum_{k=1}^{K} y_i^k log(p^k(x_i)) \tag{44}$$

où $p_k(x_i)$ correspond à la probabilité conditionnelle d'appartenir à la classe k, i.e:

$$p_k(x_i) = \frac{exp(f^k(x_i))}{\sum\limits_{k=1} K exp(f^k(x_i))} \tag{45}$$

On travaille donc à présent sur les indicatrices(y^k), et on construit des modèles (généralement des arbres) sur les gradients négatifs pour les K classes. f^k est ainsi le modèle additif issu des K modèles précédents.

A.9.1.3 *Tree Gradient Boosting*

Le Tree Gradient Boosting, proposé par Breiman, est le cas particulier du Gradient Boosting où les modèles h_m choisis sont des arbres de décision. Les paramètres à optimiser peuvent alors être les poids w_j à chaque feuille, ainsi que le nombre de feuilles T dans chaque arbre. Le XGboost présenté ci-après correspond alors à une variante permettant une implémentation plus rapide de cet algorithme.

A.9.2 *Du Gradient Boosting au XGBoost*

XGBoost [Chen and Guestrin 2016] est une manière optimisée de réaliser l'algorithme de Tree Gradient Boosting. Le construction séquentielle du Gradient Boosting le différencie des forêts aléatoires mais le rend plus lent dans sa construction car chaque itération est dépendante de la précédente. L'algorithme tente en effet d'optimiser à chaque itération une fonction objective régularisée (en norme $\mathbb{L}^1$ et $\mathbb{L}^2$), constituée d'une fonction de coût et d'un terme de pénalité pour tenir compte de la complexité du modèle. A chaque itération, un nouveau modèle est calibré sur la partie résiduelle issue de l'itération précédente. Le modèle final est alors la somme pondérées des prédictions faites par chaque sous-modèle ainsi calibrés.

L'intérêt du XGBoost réside dans la correction d'un problème majeur du Tree Gradient Boosting, à savoir le fait de considérer la perte potentielle pour chaque séparation possible (*split*) à un noeud donné pour créer une nouvelle branche. En effet, dans le

cas où nous disposons de beaucoup de variables explicatives, il y a un grand nombre de splits possibles, ce qui engendre des temps de calculs élevés.

Notons que nous allons présenter l'algorithme uniquement dans le cadre de la régression, mais celui-ci s'adapte pour la classification.

A.9.2.1 *L'algorithme*

Considérons un problème de régression pour lequel nous voulons prédire la variable $Y \in \mathbb{R}$, à partir d'un vecteur X de p variables explicatives: $X \in \mathbb{R}$ ($p \in \mathbb{N}$). Nous reprenons les idées du Boosting, à savoir agréger les prédictions de plusieurs classifieurs dits faibles pour former un classifieur fort.

ENTRAÎNEMENT SÉQUENTIEL

Soit B le nombre d'étapes de notre algorithme. Notons pour $b \in \{1, ..., B\}$:

- H l'ensemble des modèles sur lequel on va réaliser l'optimisation
- j : $\mathbb{R}$x$\mathbb{R} \to \mathbb{R}$ une fonction de coût choisie au préalable (comme le MSE par exemple).
- $\hat{h}_b : \mathbb{R}^p$ x $\mathbb{R}^n \to \mathbb{R}^n$ la fonction associée au modèle M_b de l'étape b de l'algorithme, renvoyant le vecteur de prédiction de l'input. Notons bien que le modèle M_b apprend sur les valeurs résiduelles à l'étape b et non la valeur de y.
- $\hat{y}^{(b)}$ notre estimation de y à l'étape b.
- $r^{(b)}$ le vecteur de taille n des pseudos-résidus au début de l'étape b, c'est-à-dire la partie que la partie résiduelle de la valeur cible initale qu'on a pas encore réussi à expliquer avant l'itération b: $r^{(b)} = y - \hat{y}^{(b-1)}$ et $r^{(0)} = \hat{y}^{(0)}$
- $\hat{r}^{(b)}$ l'estimation du vecteur $r^{(b)}$ faite par l'algorithme M_b à l'étape b, soit: $\hat{r}^{(b)} = \hat{h}_b(x)$.

On peut résumer ceci par les relations suivantes:

$$\hat{y}_i^{(b)} = \sum_{k=0}^{b} \hat{r}_i^{(k)} = \hat{y}_i^{(b-1)} + \hat{r}_i^{(b)} \; et \; \hat{y}_i^{(0)} = \hat{h}(x_i), \; 1 \leq b \leq B, 1 \leq i \leq n \tag{46}$$

On définit également Ω une fonction qui prend en entrée un modèle (un arbre de décision par exemple) et renvoie une mesure de la complexité du modèle. L'algorithme se présente comme ceci:

- Initialisation: on définit un modèle constant $\hat{h_0}$. Par exemple: $\hat{h_0}(x_i) = C, 1 \leq i \leq n$, avec $C \in \mathbb{R}$. Par exemple $C = 0$. On obtient $\hat{y}^{(0)} = \hat{r}^{(0)} = C$

- Pour b allant de 1 à B:
  - Calibrage du modèle M_b et de la fonction $\hat{h_b}$ associée, ajustés sur les données $(x, y - \hat{y}^{(b-1)})$ avec $y - \hat{y}^{(b-1)} = \hat{r}^{(b)}$. c'est-à-dire que $\hat{h}_b(x_i)$ donne une estimation de $y_i - \hat{y_i}^{(b-1)}, 1 \leq i \leq n$ en optimisant la fonction de coût:

$$\hat{h_b} = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} j(y_i, \hat{y_i}^{(b)}) + \sum_{l=1}^{b} \Omega(h_l) \tag{47}$$

$$= \underset{h_b \in H}{argmin} \sum_{i=1}^{n} j(y_i, \hat{y_i}^{(b-1)} + h_b(x_i)) + \Omega(h_b) + C_b \tag{48}$$

avec C_b une constante dépendant seulement des itérations avant b. Dans le cas d'une fonction de coût MSE, on obtient:

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} (y_i - (\hat{y_i}^{(b-1)} + h_b(x_i))^2 + \Omega(h_b) + C_b \tag{49}$$

Soit:

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} [2(\hat{y_i}^{(b-1)-y_i}) h_b(x_i) + h_b(x_i)^2] + \Omega(h_b) + C_b \tag{50}$$

La forme obtenue, avec un terme d'ordre 1 et d'ordre 2 en peut également être retrouvée dans le cas de fonctions de coût différentes de la MSE, en utilisant son développement de Taylor à l'ordre 2:

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(b-1)}) + \alpha_i^{(b)} h_b(x_i) + \frac{1}{2} \beta_i^{(b)} h_b(x_i)^2 + \Omega(h_b) + C_b \tag{51}$$

avec:

* $\alpha_i^{(b)} = \frac{\partial j}{\partial y}(y_i, \hat{y}^{(b-1)})$
* $\beta_i^{(b)} = \frac{\partial^2 j}{\partial y^2}(y_i, \hat{y}^{(b-1)})$

Il s'agit d'une approximation licite car les termes $h_b(x_i)$ sont supposées de taille négligeable étant donné qu'il s'agit de classifieurs faibles.
Ainsi, en supprimant toutes les constantes (dépendant des étapes précédant l'étape b), on doit résoudre le problème d'optimisation suivant:

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} \alpha_i^{(b)} h_b(x_i) + \frac{1}{2} \beta_i^{(b)} h_b(x_i)^2 + \Omega(h_b) \tag{52}$$

C'est comme ceci que XGboost fonctionne: il peut optimiser chaque fonction de perte choisie, en utilisant le même solveur en prenant les coefficients $\alpha_i^{(b)}$ et $\beta_i^{(b)}$ comme input.

- Finalement, notre estimation de y est donnée par $\hat{y}^{(B)}$: il s'agit de notre output.

COMPLEXITÉ DU MODÈLE ET RÉGULARISATION

Nous avons présenté dans l'algorithme une fonction de complexité Ω. Cette fonction contient un terme de régularisation afin d'éviter le surapprentissage. Plaçons nous dans le cas où les classifieurs faibles utlisés sont des arbres. Notons T le nombre de feuilles au dernier niveau de l'arbre, w_t le score prédit par la feuille t ($1 \leq t \leq T$). La fonction h_b associée à l'arbre de l'itération b ($1 \leq b \leq B$) est définie par : $h_b(x_i) = w_{s(x_i)}$, avec $s : \mathbb{R}^p \to \{1, ..., T\}$ une fonction renvoyant le numéro de la feuille associée au point considéré. Avec une écriture vectorielle, on a: $h_b(x) = w_{s(x)}$. La fonction de complexité du XGBoost est alors généralement définie par:

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} w_k^2 \tag{53}$$

où γ et λ sont des paramètres de régularisation à choisir.

ALGORITHME DE CONSTRUCTION DES ARBRES

En reprenant, notre formule d'optimisation: $\sum_{i=1}^{n} \alpha_i^{(b)} h_b(x_i) + \frac{1}{2}\beta_i^{(b)} h_b(x_i)^2 + \Omega(h_b)$ et en injectant notre fonction de complexité définie juste avant, on obtient:

$$\sum_{i=1}^{n} \alpha_i^{(b)} h_b(x_i) + \frac{1}{2}\beta_i^{(b)} h_b(x_i)^2 + \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} w_k^2 \tag{54}$$

Soit, en intervertissant les sommes:

$$\sum_{j=1}^{T} [(\sum_{i \in I_j} \alpha_i^{(b)}) w_j + \frac{1}{2}(\sum_{i \in I_j} \beta_i^{(b)} + \lambda) w_j^2] + \gamma T \tag{55}$$

avec $I_j = \{i \in \{1, ..., T\} | s(x_i) = j\}$ l'ensemble contenant tous les indices des points qui ont été conduits à la feuille j. En notant $A_j = \sum_{i \in I_j} \alpha_i^{(b)}$ et $B_j = \sum_{i \in I_j} \beta_i^{(b)}$, on obtient:

$$\sum_{j=1}^{T} [A_j w_j + \frac{1}{2}(B_j + \lambda) w_j^2] + \gamma T \tag{56}$$

Dans cette équation les score de prédiction w_j n'intéragissent pas entre-eux et sont donc indépendants. Pour une fonction s donnée, on peut optimiser chaque terme de la somme

indépendamment, à savoir: $A_j w_j + \frac{1}{2}(B_j + \lambda) w_j^2$.
Finalement on obtient, en excluant la solution $w_j = 0$:

$$\forall j \in \{1, ..., T\}, w_j = -\frac{A_j}{B_j + \lambda} \tag{57}$$

et une fonction objective associée vallant alors: $-\frac{1}{2} \sum_{j=1}^{T} \frac{A_j^2}{B_j + \lambda} + \gamma T$, qui mesure à quel point l'arbre associé à la fonction s, est bon (en terme de prédiction et de complexité). Il faut à présent trouver le meilleur arbre, avec la fonction de tri s la plus performante associée. Il faudrait en théorie énumérer tous les arbres possibles et choisir le meilleur mais cela demanderait trop de temps. Une méthode utilisée en pratique est une optimisation étage par étage de l'arbre. L'idée est de regarder si l'ajout de nouvelles feuilles à un niveau donné fournit un gain comparé à l'abre initial, tout en tenant compte de la régularisation. Dans le cas de l'ajout de deux feuilles à un niveau donné, il faut que le gain associé (régularisé) soit positif :

$$Gain = \frac{1}{2}[\frac{A_G^2}{B_G + \lambda} + \frac{A_L^2}{B_L + \lambda} - \frac{(A_G + A_L)^2}{B_G + B_L + \lambda}] - \gamma \tag{58}$$

L'indice G réfère à la feuille gauche ajoutée, et D à la feuille droite. Finalement il faut que le gain d'ajout de ces deux feuilles, comparé à la situation précédente, soit supérieur au coefficient γ de régularisation. Ainsi, pour construire un arbre, on cherche récursivement la meilleure division possible jusqu'à ce qu'on atteigne la profondeur maximale. Ensuite, on élague les noeuds avec un gain négatif avec une approche du bas vers le haut (bottom-up).

A.9.2.2 *Avantages du XGBoost*

Les avantages de l'algorithme XGBoost sont nombreux, et peuvent expliquer pourquoi il est tant utilisé dans les compétitions de machine learning.

- Il existe de nombreux hyperparamètres que l'on peut régler au plus fin pour optimiser ses performances. Ils permettent de garder un contrôle sur le compromis biais/variance. On peut citer notamment:
  - Paramètres de contrôle de la complexité du modèle:
    - Le nombre de sous-arbres h_m entraînés
    - La profondeur maximale de chaque abre h_m
    - Gain minimum γ requis pour autoriser une division supplémentaire
  - Paramètres de robustesse au bruit:
    - ratio de sous échantillonage de la base d'apprentissage,compris entre 0 et 1

    - ratio de sous-échantillonnage des colonnes lors de la construction de chaque arbre
  - Le taux d'apprentissage (learning rate). Il s'agit du coefficient utilisé dans la méthode de descente du gradient pour contrôler la vitesse de convergence de l'algorithme. Si il est trop bas, le temps de convergence peut être long, et si il est trop élevé, l'algorithme risque de ne pas converger.
  - Les paramètres γ et λ de régularisation $\mathbb{L}^1$ et $\mathbb{L}^2$.
  - Ajustement possible dans le cas d'une classification avec des données déséquilibrées entre les classes (en utilisant par exemple la métrique "AUC").
- Rapidité de calcul, en comparaison du Tree Gradient Boosting classique.

A.10 ALGORITHME DE CONSTRUCTION DE LA COURBRE PDP

La procédure pour générer le graphique PDP est décrit par l'algorithme suivant. Il repose sur les notations introduites dans le chapitre 6.

- Entrée : la base d'apprentissage $(x_j^{(i)})_{1\leq i\leq n, 1\leq j\leq p}$, le modèle $\hat{f}$, une variable à expliquer supposée être x_1 ici pour simplifier, i.e. $S = 1$.
  c'est-à-dire que : $(x_j^{(i)})_{1\leq i\leq n, 1\leq j\leq p} = (x_1^{(i)}, x_C^{(i)})_{1\leq i\leq n}$
- Pour i allant de 1 à n:
  - Copie de la base d'apprentissage, en remplaçant la valeur de la variable x_1 par la valeur constante $x_1^{(i)}$: $(x_1^{(i)}, x_C^{(k)})_{1\leq k\leq n}$
  - Calcule du vecteur de prédiction par $\hat{f}$ des données précédemment définies : $\hat{f}(x_1^{(i)}, x_C^{(k)})$ pour $k = 1, ..., n$
  - Calcul de $\hat{f}_{x_1}(x_1^{(i)})$ par la formule: $\hat{f}_{x_1}(x_1^{(i)}) \simeq \frac{1}{n}\sum_{k=1}^{n} \hat{f}(x_1^{(i)}, x_C^{(k)})$
- Sortie : le graphique des points $(x_1^{(i)}, \hat{f}_{x_1}(x_1^{(i)}))$ pour $i = 1, ..., n$, appelé graphique de dépendance partielle (PDP).

A.11 PRÉCISION SUR LA MÉTHODE ICE

L'approche par les courbes ICE fournit un graphique avec une ligne pour chaque instance, qui montre comment la prédiction change quand une caractéristique change. A la place de la moyenne réalisée lors du calcul de la PDP, le calcul de l'ICE est réalisé pour chaque instance. Ainsi, on obtient un graphique ICE, contenant autant de courbes que d'observations. Cette méthode a été introduite par Goldstein (2014). Contrairement au graphique de dépendance partielle qui est une approche globale, les courbes ICE sont locales (c.f 94). L'algorithme utilisé pour estimer l'ICE est le suivant:

- Entrée: les données d'entraînement: $(x_j^{(i)})_{1\leq i\leq n, 1\leq j\leq p}$, le modèle ajusté $\hat{f}$, S un sous-ensemble de $\{1, ...p\}$ et C le complémentaire de S dans $\{1, ..., p\}$.
- Pour i=1,...,n:
  - $\hat{f}_S^{(i)} = 0_{n\times 1}$
  - $x_C = x_C^{(i)}$: on fixe les colonnes C à la i-ième observation.
  - Pour l=1,...n:
    - $x_S = x_S^{(l)}$
    - $\hat{f}_{S,l}^{(i)} = \hat{f}(x_S, x_C)$
- Output: $\hat{f}_S^{(1)} = (\hat{f}_{S,1}^{(1)}, ..., \hat{f}_{S,n}^{(1)}), ..., \hat{f}_S^{(n)} = (\hat{f}_{S,1}^{(n)}, ..., \hat{f}_{S,n}^{(n)})$

Reprenons l'exemple de la partie sur le graphique de PDP page 245. Dans celui-ci, on a observé que la PDP, en réalisant une moyenne, ne capte pas toute la dépendance d'une variable sur la prédiction. Affichons à présent le graphique d'ICE associé à la PDP. Cette fois, l'ICE capte l'effet de X_2 sur la prédiction pour chaque instance. On observe

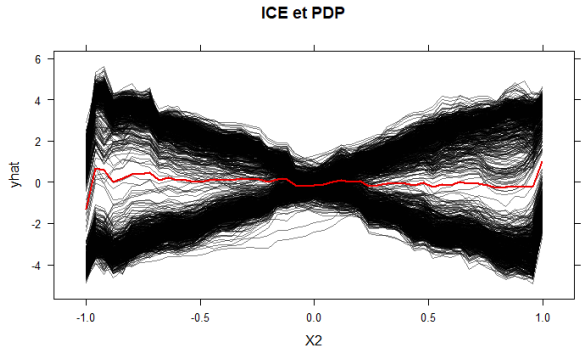


Figure 109.: Graphique de PDP (en rouge) et ICE (en noir)

qu'en moyenne le graphique d'ICE est proche de 0, ce qui correspond à la PDP, mais que le graphique d'ICE complet est proche du nuages de points représenté sur la figure 96. Il existe une adaptation de la méthode ICE, appelée c-ICE, permettant de centrer les courbes et ainsi de mieux voir les effets de chaque variable sur la prédicition. D'autres variantes qui s'appuient sur les dérivées partielles telles que le d-ICE existent également.

Les courbes ICE présentent globalement les mêmes intérêts que le graphique PDP, notamment la simplicité d'implémentation et d'interprétation. L'autre avantage important est qu'elles ne masquent pas les effets hétérogènes du modèle considéré. Ainsi, en utilisant le graphique PDP, qui fournit un résumé de l'impact d'une variable sur la prédiction du modèle, et les courbes ICE, qui le précisent, on obtient une bonne explication

globale des prédictions.

Cependant, tout comme le PDP, les courbes ICE reposent sur une hypothèse d'indépendance entre les différentes variables, et ne tient pas compte de leur distribution réelle. Un autre inconvénient est le fait de ne pouvoir représenter ses courbes qu'en 2 ou 3 dimensions, du fait que l'humain ne sait pas se représenter des dimensions supérieures. De plus, le graphique contenant toutes les courbes ICE est vite surchargé lorsque le nombre d'individus étudié est grand.

A.12 PRÉCISION SUR LA MÉTHODE GRAPHIQUE ALE

A.12.0.1 *Formalisme mathématique*

Pour bien comprendre la méthode ALE, nous nous appuyons sur la formule de la méthode PDP. La PDP associée aux variables X_S repose sur le calcul suivant :

$$\hat{f}_{X_S,PDP}(x_S) = \mathbb{E}_{X_C}[\hat{f}(x_S, X_C)] = \int_{x_C} \hat{f}(x_S, x_C)\mathbb{P}_{X_c}(x_C)dx_C \tag{59}$$

calcul mené sur chaque point x_S de la distribution marginale de la variable X_S. Le M-plot, lui repose sur le calcul de la moyenne des prédictions sur la distribution conditionnelle, à savoir :

$$\hat{f}_{X_S,M-plot}(x_S) = \mathbb{E}_{X_C|X_S}[\hat{f}(X_S, X_C)|X_S = x_S] = \int_{x_C} \hat{f}(x_S, x_C)\mathbb{P}(x_C|x_S)dx_C \tag{60}$$

Finalement, le graphique ALE nécessite la définition d'une borne $z_{0,1}$ pour délimiter l'intervalle sur lequel la moyenne des différences de prédiction est calculé. On obtient alors :

$$\hat{f}_{X_S,ALE}(x_S) = \int_{z_{0,1}}^{x_S} \mathbb{E}_{X_C|X_S}[\hat{f}^{(S)}(X_S, X_C)|X_S = x_S]dz_S = \int_{z_{0,1}}^{x_S}\int_{x_C} \hat{f}^{(S)}(x_S, x_C)\mathbb{P}(x_C|x_S)dx_C dz_S \tag{61}$$

où $\hat{f}^{(S)}(x_S, x_C) = \frac{\partial \hat{f}(x_S,x_C)}{\partial x_S}$ est le gradient.

A.12.0.2 *Estimation de l'ALE*

En pratique la construction se fait de manière discrète. Par exemple, considérons une variable numérique x_j ($S = \{j\}$) où $j \in \mathbb{N}$. Soit $x \in \mathbb{X}$, on divise en pratique la variable x_j en plusieurs intervalles, à savoir: $[z_{0,j}, z_{1,j}]$, $[z_{1,j}, z_{2,j}]$,...,$[z_{k_j(x)-1,j}, z_{k_j(x),j}]$, avec $k_j(x)$ le nombre d'intervalles et $z_{0,j} < z_{1,j} < ... < z_{k_j(x),j}$. Pour $k \in \{1, ..., k_j(x)\}$, on note $N_j(k)$ l'ensemble des individus de la base d'apprentissage pour lesquels la variable x_j est

dans le $k^{ième}$ intervalle : $[z_{k-1,j}, z_{k,j}]$, et $n_j(k)$ le nombre d'individus dans $N_j(k)$. Alors l'ALE au point $x \in \mathbb{X}$ associée à la variable j est estimée par la formule:

$$\hat{\tilde{f}}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \Big[f(z_{k,j}, x_{\setminus j}^{(i)}) - f(z_{k-1,j}, x_{\setminus j}^{(i)}) \Big] \tag{62}$$

Le terme d'effets locaux accumulés s'expliquent par cette formule : sur chaque intervalle on mesure la différence de prédiction "locale" puis on somme sur tous les intervalles, afin d'avoir l'effet "accumulé". En réalité, la véritable définition de l'ALE centre le terme précédent afin d'avoir un effet moyen à 0. Il en découle la formule suivante:

$$\hat{f}_{j,ALE}(x) = \hat{\tilde{f}}_{l,ALE}(x) - \frac{1}{n} \sum_{l=1}^{n} \hat{\tilde{f}}_{l,ALE}(x) \tag{63}$$

$$= \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \Big[f(z_{k,j}, x_{\setminus j}^{(i)}) - f(z_{k-1,j}, x_{\setminus j}^{(i)}) \Big] - \frac{1}{n} \sum_{l=1}^{n} \hat{\tilde{f}}_{l,ALE}(x) \tag{64}$$

Le fait de centrer la formule permet d'interpréter l'ALE comme l'effet d'une variable sur la prédiction en comparaison de la prédiction moyenne sur la base de données d'apprentissage. Ainsi, si on obtient une ALE de -2 à un certain point x lorsque $x_j = 3$, cela signifie que lorsque la j-ième variable vaut 3, alors la valeur de prédiction est inférieure de 2, en comparaison de la prédiction moyenne.

Les intervalles dans la formule de l'ALE 62 sont souvent choisis comme différents quantiles de la distribution de la variable x_j qu'on étudie. Cela permet d'avoir autant d'individus dans chaque intervalle, mais présente l'inconvénient d'avoir des intervalles de tailles très variables, notamment si la queue de la distribution est lourde.

A.13 LIMITES DE LIME ET NOUVELLE MÉTHODE LS

Dans cette partie, nous reprenons les concepts introduits dans l'article [**?** **?**] *Defining Locality for Surrogates in Post-hoc Interpretablity*, qui montre les limites de LIME et propose une nouvelle méthode.

Les approches par modèles de substitution locaux (*local surrogates*) font parties des méthodes pour comprendre les prédictions faites par un modèle (souvent complexe qu'on associe à une "boite noire").

L'article [T. Laugel 2018] souligne l'importance de définir la "bonne" localité, c'est-à-dire le voisinage sur lequel le modèle de substitution local va apprendre, afin d'approcher

précisément les limites de décision locales de la boîte noire. Le problème souligné dans cet article, est qu'il ne s'agit pas uniquement d'un problème de paramètre ou de distribution de l'échantillon. Cela a donc un impact sur la pertinence et la qualité de l'approximation des limites de décision locales de la boîte noire et ainsi sur le sens et la précision des explications fournies par les méthodes comme LIME. Pour surmonter les problèmes identifiés, quantifiés avec une mesure et une procédure adaptées, l'article propose de générer des explications, toujours basées sur des modèles de substitution, avec des échantillons centrés sur un endroit particulier de la limite de décision (boundary limit), utile pour la prédiction à donner, plutôt que sur la prédiction elle-même comme c'est classiquement fait. Cette nouvelle approche est ensuite comparée aux méthodes habituelles et apporte une amélioration sur la qualité d'interprétation fournie.

On considère un modèle de type "boîte noire" (SVM, Random Forest...): $b : \mathbb{X} \to \mathbb{Y}$. On souhaite expliquer la prédiction $b(x)$ du modèle pour une instance $x \in \mathbb{X}$. Pour se faire, on met en place un modèle de substitution local s_x (surrogate local model) pour expliquer les limites de décision (decision boundary) de b. Pour se faire, il existe 3 étapes:

- Le choix d'une base d'apprentissage (X_{s_x}, Y_{s_x}) pour s_x, à partir de la base d'apprentissage initiale B_a.
- A partir de cet échantillon l'ajustement du modèle s_x peut-être mis en place dans le but d'expliquer le comportement local de b autour de l'instance x. On peut, par la suite, ajouter des contraintes pour contrôler la complexité du modèle de substitution ou sa localité.
- Utilisation du modèle s_x pour expliquer la prédiction $b(x)$ faite par le modèle boîte noire.

Il se pose alors la question du choix et de la manière de générer la base d'apprentissage X_{s_x} de modèle de substitution. On pourrait utiliser directement la base d'apprentissage utilisée pour entraîner b, mais il a été montré par Craven et Shavlik en 1996 qu'augmenter localement la densité des instances est bénéfique pour la précision du modèle de substitution. Ainsi, il nous faut trouver une autre base d'apprentissage. Etant donné que l'on cherche un modèle local, il nous faut définir un voisinage. Analysons le choix fait par LIME et proposons ensuite une alternative.

A.13.0.1 *LIME et le choix du voisinage d'un modèle de substitution local*

La procédure de LIME pour expliquer la prédiction faite par b pour l'instance x repose sur les étapes détaillées dans l'algorithme :

- La base d'apprentissage X_{s_x} pour calibrer s_x est construite en simulant des échantillons d'une loi normale pour chaque variable explicative x_i, avec la même moyenne

et le même écart-type que les variables utilisées pour ajuster b. On note pour tout $j \in \{1, ..., p\}$, $\mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$ (moyenne empirique de la variable x_j dans la base d'apprentissage initiale) et $\sigma_j^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \mu_j)^2$ (variance empirique de la variable x_j dans la base d'apprentissage initiale). On note aussi n_{sim} le nombre d'individus utilisés pour ajuster s_x. Alors pour tout $j \in \{1, ..., p\}$, on simule n_{sim} échantillons indépendants de loi $N(\mu_j, \sigma_j^2)$. Ceci forme notre base X_{s_x}. Dans le cas de variables catégorielles, LIME travaille avec les probabilités de prédiction renvoyées par b.

- On ajuste le surrogate s_x en utilisant un modèle linéaire avec régularisation de type ridge, c'est-à-dire qu'on ajoute un paramètre de pénalité pour contrôler la variance du modèle. Chaque instance $\tilde{x}$ de X_{s_x} se voit attribuer un poids calculé par rapport à sa distance avec x, à l'aide d'une fonction noyau comme le noyau RBF défini par: $RBF(x, \tilde{x}) = exp(-\frac{||x-\tilde{x}||^2}{2\sigma^2})$, avec $\sigma > 0$ un paramètre à définir. Cette pondération a pour objectif de favoriser les points proches du point x que l'on cherche à expliquer, afin d'avoir une interprétation locale.

- On génère des explications de la boîte noire b en extrayant les coefficients de la régression linéaire mise en place avec s_x.

T. Laugel (2018) montrent alors que l'approche mise en place par LIME pour définir son voisinage a tendance à cacher les *features* avec une influence locale au profit des *features* avec une influence globale. Ceci a été mis en évidence sur le dataset "half moon" disponible dans le package "scikit-learn" de Python. C'est suite à cette découverte qu'une nouvelle approche, dans la continuité de LIME, a été introduite. Pour se faire, critère numérique pour évaluer la localité d'un point est introduite. On définit, à partir d'une mesure de précision Acc (telle que le score AUC), la fidélité locale $LocFid$ d'un modèle de substitution s_x pour la boîte noire b dans le voisiniage V_x autour de l'instance x que l'on souhaite définir: $LocFid(x, s_x) = \underset{x_i \in V_x}{Acc} (b(x_i), s_x(x_i))$. Il nous faut à présent définir le voisinage V_x choisi. L'article suggère que ce soit l'hypersphère l_2 centrée sur x de rayon $r_{fid} > 0$. Ceci permet de considérer r_{fid} comme un proxy du degré de localité considéré. Etant donné que la valeur du rayon est étroitement liée à la dimension et à la densité de l'espace de départ $\mathbb{X}$, r_{fid} sera exprimé comme un pourcentage de la distance maximale entre les instances de la base de données initiales et l'instance x que l'on souhaite interpréter. L'étude a montré qu'avec LIME, la fidélité à l'échelle locale (r_{fid} petit) est moins bonne qu'à l'échelle globale (r_{fid} élevé), ce qui confirme ce qui a été dit précédemment. Afin de remédier à ce problème de fidélité à l'échelle locale, l'idée est d'ajuster le modèle de substitution sur un voisinage de la limite de décision à approcher, plutôt que sur un voisinage de l'instance x que l'on cherche à expliquer. En effet, LIME réalise l'ajustement du modèle de substitution sur toutes les données

de la base d'apprentissage, en pondérant chaque instance par rapport à sa distance à l'instance x à expliquer.

A.13.0.2 *Méthode LS (Local Surrogate)*

La méthode LS (Local Surrogate) utilise la remarque faite précédemment. Son principe est très similaire à LIME mais diffère dans la première étape. En effet, la méthode LS calcule tout d'abord l'instance x_{bord}, la plus proche de l'instance à expliquer x telle que: $b(x) \neq b(x_{bord})$. Pour la calculer, la partie "Génération" de l'algorithme *GrowingSpheres* introduit par Laugel et al. (2018). Le principe est de générer des instances dans une hypersphère de rayons croissants, centrées sur x, jusqu'à atteindre la limite de décision de b. Une fois que x_{border} a été trouvé, les instances pour ajuster le modèle local sont tirées uniformément dans une hypersphère S de rayon r_{s_x}, centrée sur x_{bord}: $X_{s_x} \sim \mathbb{U}_{S(x_{bord}, r_{s_x})}$. On peut résumer la procédure LS par les étapes détaillées dans l'algorithme :

- Entrée : $x \in \mathbb{X}$, $b : \mathbb{X} \to \mathbb{Y}$, $r_{s_x} \in \mathbb{R}$, $N \in \mathbb{N}$
- Calculer de x_{bord} à l'aide de l'algorithme GrowingSpheres avec le modèle b et l'instance x
- Calcul de la base d'apprentissage X_{s_x} pour ajuster le modèle de substitution: tirage de N instances indépendantes dans l'hypersphère de rayon x_{bord} de rayon r_{s_x}.
- Calculer de la sortie de la boîte noire sur la base X_{s_x}: $Y_{s_x} = b(X_{s_x})$.
- Entraîner le modèle de substitution sur la base (X_{s_x}, Y_{s_x})
- Sortie: le modèle de substitution s_x