Some statistical learning problems with incomplete data

Maximilien Baudry

N°d'ordre NNT : 2020LYSE1002

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° 512
Ecole Doctorale Informatique et Mathématiques

Spécialité de doctorat : Mathématiques appliquées

Soutenue publiquement le 08/01/2020, par :
Maximilien Baudry

Quelques problèmes d'apprentissage statistique en présence de données incomplètes

---

Devant le jury composé de :

Julie Josse	Présidente
Thierry Artières	Rapporteur
Olivier Lopez	Rapporteur
Gérard Biau	Examinateur
Anne-Laure Fougères	Examinatrice
Christian Y. Robert	Directeur de thèse
Sébastien Conort	Invité

Quelques problèmes d'apprentissage statistique en présence de données incomplètes

Maximilien Baudry

Laboratoire de Sciences Actuarielle et Financière - EA 2429
Université Claude Bernard Lyon 1

Thèse pour l'obtention du grade de :
Docteur de l'université Claude Bernard Lyon 1

Sous la direction de : Christian Y. Robert

Rapportée par : Thierry Artières
Olivier Lopez

Présentée devant un jury composé de : Julie Josse *(présidente)*
Thierry Artières *(rapporteur)*
Olivier Lopez *(rapporteur)*
Gérard Biau *(examinateur)*
Anne-Laure Fougères *(examinatrice)*
Sébastien Conort *(invité)*
Christian Y. Robert *(directeur)*

École Doctorale InfoMaths de Lyon
Section Mathématiques appliquées

Mercredi 8 janvier 2020

Remerciements

Trois ans. Trois ans se sont écoulés et j'ai l'impression d'avoir commencé ma thèse hier. Lorsque je prends du recul, je n'arrive pas à croire tout ce que j'ai pu apprendre entre le début et la fin de cette thèse. Tout cela je le dois à chacun d'entre vous qui êtes dans ces remerciements. Et toi que j'ai oublié, sache que je suis profondément désolé, et que je te dois un verre pour ça.

Impossible de commencer par quelqu'un d'autre que Christian Y. Robert, mon directeur de thèse, mentor et ami. Christian, je ne pouvais tomber sur une meilleure personne que toi pour encadrer ma thèse. Ton côté humain, ton sens de l'écoute et de la communication et ton incroyable intuition mathématique font de toi une personne exceptionnelle, et je suis fier d'avoir été ton élève. Christian est une personne (très !) exigeante, qui a su s'adapter à ma manière de travailler, m'écouter et aussi me botter les fesses quand il le fallait. Merci à toi Christian.

Je tiens également à remercier chaleureusement Thierry Artières et Olivier Lopez d'avoir rapporté ma thèse, j'espère être à la hauteur de vos exigences.
Je remercie également Gérard Biau, Julie Josse et Anne-Laure Fougères de faire partie de mon jury en tant qu'examinateurs, c'est un honneur pour moi d'être devant vous aujourd'hui.

Un merci tout particulier à Julien Velcin pour avoir bien voulu faire partie de mon comité de suivi de thèse avec Olivier Lopez.

Un grand grand grand merci à Sébastien Conort, d'une part pour avoir accepté de faire partie de mon jury, et d'autre part pour m'avoir acueilli et fait partager le quotidien de tes équipes. Je me suis senti intégré et faisant partie de l'équipe dès les premiers jours de ma thèse, et nous en avons vécu des choses ! Je te remercie profondément, pour ton humanisme, ton écoute et ton exigence qui ont su me challenger tout au long de la thèse.

Je remercie également les membres de BNP Paribas Cardif, pour m'avoir soutenu au quotidien pendant tout ce temps, notamment je remercie Michael de Toldi, Jean-Paul Felix, Bernard Bolle Reddat, et la bande de passionnés de l'équipe de Sébastien avec qui j'ai eu le privilège de partager mon quotidien, notamment Lam Dang (une véritable machine à idées); Philippe Baudier (avec qui j'ai eu de longues et passionnantes discussions sur la psychologie); Jean-Baptiste Foncin (qui s'est acheté une véritable machine de guerre pour faire des compétitions Kaggle, mais aussi pour chauffer son appartement). Je remercie également Youcef Kacer, Damien Hennom, Imad Amrouss, Taycir Yahmed et Ahmed Sahraoui, avec qui j'ai moins eu l'occasion d'échanger, mais qui sont habités par une passion commune.
Je remercie également les anciens membres du DataLab de BNP Paribas Cardif, notamment Guillaume Huard et Paul Todorov, avec qui nous avons gagné un challenge Kaggle avec Sébastien et Lam, expérience inoubliable qui a débuté lors d'une conférence en Machine Learning à Toulon, et je remercie également Joseph Assouline, qui m'a grandement aidé à comprendre des concepts important pour l'une des contributions de cette thèse.
En parlant du DataLab de Cardif, impossible de ne pas remercier Folly Akakpovi et Florian Silva, et surtout le maître, l'incroyable, le magnifique Anh Vinh Nguyen Ton (non rien).

Enfin, une dernière personne de chez Cardif que je voudrais chaleureusement remercier est Anat Chefner, avec qui nous avons eu de passionnantes discussions de Machine Learning, et je me souviens encore de la délivrance et la joie que l'on a eu lorsque l'on a réussi à résoudre ce maudit bug (Anat reconnaîtra ledit bug ahah).

Je voudrais également remercier les personnes qui m'ont beaucoup aidé administrativement du côté de la chaire DAMI, notamment Teddy Arrif et surtout Sophie Castelbou qui s'est démenée contre vents et marées, pour réserver mes billets de train, gérer les réservations d'hôtel ou encore organiser les évènements de la chaire, le tout en restant pétillante et en gardant le sourire, merci à toi Sophie.

Je tiens à remercier le laboratoire SAF pour m'avoir réservé le meilleur accueil tout au long de ces années bien que j'aie été présent par morceaux au cours du temps. Un merci tout particulier aux doctorants lyonnais, notamment Sarah Ben Salem, alias Mastermind, qui a su me remonter le moral dans les moments difficiles et avec qui j'ai pu exposer la vie de mon chat en toute liberté ; Steve Briand, alias le BG, s'il te

plait ne cuisine plus jamais pour moi, jamais ; Morgane Plantier, alias la patronne du bureau 2410, son autorité nous a permis de survivre à la pénurie de café l'été dernier ; et Romain Gauchon, le maître des énigmes, qui nous a partagé sa culture geek, jap, culinaire, une encyclopédie sur pattes !
Merci également aux autres doctorants et post-doc, Patrick Laub, Pierre Montesinos, Arthur Maillart, Pierrick Piette, Baptiste Dieltiens, même si nous avons moins eu l'occasion d'échanger, vous avez une place dans mon cœur.

Une personne que je voudrais profondément remercier est Anani Olympio, mon ancien responsable chez CNP Assurances, qui m'a fortement encouragé à me lancer dans cette thèse, et qui a su me remonter le moral dans un moment difficile de ma thèse. Merci à toi Anani.

L'aventure CNP ne s'arrête pas là, j'y ai fait des rencontres formidables, notamment avec Romain Ayres, mentor technique qui m'a appris énormément de choses, ainsi que Pierre Courtiol et Nicolas Cavallo, à qui je dois encore environ 1 m^3 de Coca à cause d'un pari que j'ai perdu.

Je remercie également du fond du cœur mes amis nerds, avec qui j'ai pu partager des soirées endiablées de Magic et de jeux de société. Notamment, je tiens à remercier Romain Mismer, le maître incontestable de la construction de decks Magic janks ET compétitifs, et Guillermo Durand, qui connaît mieux les règles de Magic que Yoda ne connaît la force, ainsi que leur compagnes Cécile et Pauline pour nous avoir supportés tout ce temps. Merci également à Arthur Nunge, imbattable à Scythe, Hadrien de March pour les longues discussions de startuppers que nous avons pu avoir en jouant à Magic et Arnaud de Milleret pour ses imitations des Gungan pendant les AP Magic.

Je remercie également les autres collègues de bureau de Léa, notamment Yating Liu, pour m'avoir cuisiné les meilleurs raviolis que j'ai pu manger et pour son fin palais pour le champagne, ainsi que son mari Yassir. Merci également à Thibaut Montes, qui n'a toujours pas trouvé comment me battre au babyfoot, et pour me ramener près de 2 kg de comté quand il rentre chez ses parents.

Merci à mes amis de l'ISUP avec qui j'ai gardé le contact, notamment Virginie Simon, et mes fidèles compagnons de Hellfest Adrien Bichot, Gabriel Barbieri et Adrien Presle (hâte de vous y retrouver l'an prochain).

Merci également à un ami très proche rencontré à Reims, Serge Masson, et à sa petite famille, pour m'avoir soutenu à distance tout ce temps et pour m'avoir toujours encouragé dans les moments difficiles de ma vie.

Les deux personnes que je voudrais remercier à présent sont Joseph Achoury Klejman, la seule personne capable de manger 2 kg de fromage lors d'une raclette (oui, oui, il l'a fait, jusqu'à "en avoir la flemme de manger"), et Valentin Prunier, fondateur avec Joseph du magnifique groupe Logos. A noter que Joseph n'ose plus jouer à Magic contre moi depuis sa défaite cuisante.

L'aventure de la thèse n'est pas la seule aventure que j'ai vécu pendant ces trois années. En effet, je tiens à remercier très profondément mes amis et associés Pierre de Sahb et Jesse Créange, avec qui j'ai co-fondé UNiFAi, pour leur patience, leur soutien et leurs encouragements tout au long de ces années. Je vous remercie Pierre et Jesse pour cette formidable aventure dans laquelle nous nous sommes lancés ensemble, votre côté humain et vos valeurs rendent le quotidien passionnant, amusant et enrichissant. Merci également à Clément Jacquemaire, alias kalak, qui nous a rejoint il y a presque un an maintenant, avec qui j'éprouve énormément de plaisir à travailler au quotidien. Tes qualités humaines font de toi quelqu'un d'unique kalak, et nous sommes chanceux de t'avoir.

L'aventure de startupper, c'est aussi une aventure collective partagée au quotidien avec d'autres entrepreneurs. Je tiens donc à remercier tout particulièrement Amaury Delagarde et Thibaut van der Werf, fondateurs de TOULEP (Tulip pour les intimes), pour les moments d'échanges que nous avons partagés, souvent autour d'un DIJOUNI. Merci également à Arnault Daubresque, pour avoir réveillé en moi le compétiteur né que je suis, en me mettant des fessées monumentales à Smash. Un jour crois moi, je me délecterai de ta défaite (ou pas).
Je remercie également le staff de l'incubateur Matrice dans lequel nous sommes, notamment Jean Condé et Thibaud Dumas, tous deux docteurs, pour m'avoir épaulé et soutenu moralement sur la fin de ma thèse, et merci à François-Xavier Petit, directeur de Matrice et mâconnais tout comme moi !

Je tiens à remercier de tout cœur la famille de Léa pour leur soutien sans faille et leurs encouragements tout au long de la thèse. Notamment, merci à Catherine Weill

et Stéphane Longepierre, Julien, Anna et Nicolas, ainsi que Charlotte Weill et Henri Wallard, qui m'a beaucoup encouragé à démarrer la thèse à l'époque, merci à toi Henri tu as bien fait !
Un merci tout particulier également à Clément Weill, qui a grandement participé à nos soirées de nerds, aussi bien à Magic qu'aux autres jeux de société, avec une énergie et un sourire toujours au rendez-vous. On n'oubliera pas également tes super performances aux AP Magic les lendemains de soirées, c'est remarquable.

Un merci tout particulier à une femme qui m'a aidé à un moment très délicat de ma vie, sans qui je n'aurais pu terminer mes études. Merci à Paule Vergnes, pour sa gentillesse et sa générosité, sans qui rien de ce que vous êtes en train de lire n'existerait.
Je voudrais également remercier madame Bellini et madame Vacher, deux professeurs de mathématiques que j'ai eues au lycée, qui ont éveillé en moi la curiosité, et surtout, surtout, la rigueur des mathématiques, merci à vous, c'est grâce à vous si j'ai continué dans cette voie.

Je remercie animalement mon chat Nessie, la gentillesse incarnée, l'incarnation même de la douceur, qui m'a beaucoup aidé à sa façon au quotidien. Merci également au chat des parents de Léa, Kitty, qui ne cesse de réclamer des câlins à chaque fois que je lui rends visite.

La FAC Team. La FAC Team, c'est ce petit groupe d'amis de l'ISUP, incollables sur les Simpson (d'ailleurs il y a bien longtemps que nous ne nous sommes pas fait un Simpsons day), et surtout un groupe d'amis soudé, qui se voit toutes les semaines pour parler de ses petits malheurs du quotidien. Merci à vous pour votre soutien profond, votre écoute, et votre capacité à me faire oublier tous mes soucis à chaque fois que l'on passe une soirée ensemble.
Merci à toi Marion, pour les raclettes au mois de juillet, pour les barbecues au mois de décembre (bien que ça soit une dangereuse idée hein), et surtout pour ce feuilleton hiérarchique que tu nous partages chaque fois qu'on se voit.
Merci à toi Kevin, alias kiki, alias kikouin the motherfucking T-Rex, ta passion pour le cinéma me fait découvrir des choses magnifiques, autant de navets (Philippe, je sais où tu te caches !) que de chefs-d'œuvre (merci pour The Wire), I love you more than a fat kid loves cake.
Merci à toi Adrien, alias Adibou, pour ces parties d'échecs palpitantes, tu es un redoutable adversaire et une source de motivation pour moi pour approfondir la théorie

des échecs, merci également pour les anecdotes que tu nous partages dans une prose magnifiquement exécutée, et surtout fais attention à bien prendre tes affaires quand tu pars de soirée ! Merci à vous les amis, je vous aime.

Rien de tout cela ne serait possible sans le soutien de notre famille. Ma famille, c'est pour elle que j'ai fait tout ce que j'ai accompli jusqu'à aujourd'hui, c'est pour elle que je me bats au quotidien, et ces personnes représentent ce que j'ai de plus cher sur cette planète. Merci à mes frères et soeur Olivier, Béatrice et Thomas, pour leurs soutien sans faille, pour leurs encouragements et pour l'amour que nous avons. Nous sommes soudés et sommes la plus belle réussite de notre mère, et rien ni personne ne pourra rien y changer. Merci à vous la fraterie, je vous aime.
Merci également à mes magnifiques neveux et nièces, Valentin, Elisa, Titouan et Maëlann.
Merci à Samuel, qui depuis que je suis tout petit me fait baigner dans les mathématiques, celui qui a éveillé ma passion pour la discipline, celui qui m'a toujours encouragé et qui a toujours su trouver les mots pour m'aider. Merci également à ta femme Christine, qui s'est occupé de moi étant petit (encore désolé de t'avoir vomi dessus quand tu m'as pris dans tes bras, c'est de la faute de Sam, j'avais déjà peur de l'avion à l'époque), et qui m'a toujours soutenu, aimé, encouragé, merci à vous deux. Je remercie également vos trois magnifiques filles, mes cousines préférées Scylla, Léna et Anaëlle, avec qui nous sommes si complices depuis toujours.
Merci à Sophie, alias la dame en noir, sur qui j'ai toujours pu compter, qui m'a toujours encouragé et soutenu, et qui m'a fait boire ma première bière.
Merci à Gérard, qui a toujours su me remettre à ma place de petit con que je suis, mais sur qui j'ai toujours pu compter et qui m'aidait à comprendre le fond des choses en maths quand j'avais besoin d'aide. Merci également à Sylvie, pour ton soutien et tes encouragements, et également pour cette recette de tarte au sucre dont je suis dingue !
Merci à Julien et Alexis, mes cousins, compagnons de Hellfest et passionnés de metal tout comme moi, merci également à Caroline, ainsi que Jean-Marie et Dominique, chez qui j'avais l'habitude de passer pendant les vacances étant petit.

Léa, merci de ton soutien durant ces trois années, dans les bons et les mauvais moments, tu as su trouver la patience de me supporter, les mots pour me calmer, et les surprises pour me faire plaisir (encore merci pour ce billet pour aller voir Rammstein). Nous partageons nos difficultés, et nous soutenons dans nos thèses respectives, et je t'en suis infiniment reconnaissant. J'ai hâte de continuer à remplir d'équations notre

joli tableau, je le trouve un peu nu ces derniers temps. Voilà maintenant presque 5 ans que j'ai la chance de partager ma vie avec toi, merci pour tout, je t'aime.

Enfin, je tiens à clore ces remerciements par les deux personnes qui me sont les plus chères. Ma mère Nicole et Daniel mon beau père. Daniel, nos débuts n'ont pas été faciles (tu te rappelles de ton surnom ?), mais tu m'as toujours considéré comme ton propre fils, tu m'as aidé, soutenu, encouragé aimé, inconditionnellement, depuis que l'on se connaît. Tu es la seule personne à avoir rendu ma mère heureuse, et pour cela, je t'en serai a jamais reconnaissant. Je te considère comme mon propre père, le cœur sur la main, toujours prêt à rendre service, et je ne connais nulle autre personne qui aime la crème autant que toi. Merci à toi.
Maman, je veux que tu saches que même si je n'ai pas été un garçon facile à élever (oh que non), tu as fait un travail absolument formidable, tu as élevé quatre enfants soudés, qui partagent les mêmes valeurs et qui s'aiment, quoi de mieux pour une mère ? Tu m'as toujours soutenu, épaulé, nous avons une complicité sans égal, nous nous comprenons d'un regard, et c'est grâce à toi si je peux présenter ce document aujourd'hui. Merci.

Je dédie cette thèse à mes parents, Nicole Paté et Daniel Bergeret.

Abstract

Most statistical methods are not designed to directly work with incomplete data. The study of data incompleteness is not new and strong methods have been established to handle it prior to a statistical analysis. On the other hand, deep learning literature mainly works with unstructured data such as images, text or raw audio, but very few has been done on tabular data. Hence, modern machine learning literature tackling data incompleteness on tabular data is scarce. This thesis focuses on the use of machine learning models applied to incomplete tabular data, in an insurance context. We propose through our contributions some ways to model complex phenomena in presence of incompleteness schemes, and show that our approaches outperform the state-of-the-art models.

Résumé

La plupart des méthodes statistiques ne sont pas nativement conçues pour fonctionner sur des données incomplètes. L'étude des données incomplètes n'est pas nouvelle et de nombreux résultats ont été établis pour pallier l'incomplétude en amont de l'étude statistique. D'autre part, les méthodes de deep learning sont en général appliquées à des données non structurées de type image, texte ou audio, mais peu de travaux s'intéressent au développement de ce type d'approche sur des données tabulaires, et encore moins sur des données incomplètes. Cette thèse se concentre sur l'utilisation d'algorithmes de machine learning appliqués à des données tabulaires, en présence d'incomplétude et dans un cadre assurantiel. Au travers des contributions regroupées dans ce document, nous proposons différentes façons de modéliser des phénomènes complexes en présence de schémas d'incomplétude. Nous montrons que les approches proposées donnent des résultats de meilleure qualité que l'état de l'art.

Contents

Résumé détaillé

La plupart des méthodes statistiques ne sont pas adaptées pour nativement tenir compte de l'incomplétude des données, et attendent en général en entrée un ensemble d'observations indépendantes et identiquement distribuées (i.i.d.). Le terme de données incomplètes réfère souvent dans la littérature aux données manquantes, néanmoins nous considérons dans ce document que cela fait référence à un ensemble de phénomènes plus large, dont les données manquantes font partie. Dans le panel de schémas d'incomplétude de données, on citera notamment les phénomènes de censure, très présents en assurance et dans les études cliniques, les données manquantes ou encore les données bruitées, pour n'en citer que quelques uns.
Par ailleurs, les travaux en deep learning se concentrent principalement sur l'étude de données non structurées, comme les images, le texte, les vidéos ou encore le son, mais très peu de travaux approfondis appliqués sur des données tabulaires ont été proposés, et encore moins sur des données tabulaires incomplètes.

Les problèmes d'incomplétude de données sont omniprésents en pratique, et peuvent conduire à un biais significatif dans l'étude statistique. Une attention particulière est donc à porter en amont de toute étude statistique afin de s'assurer de la cohérence du jeu de données observé, et éventuellement effectuer un traitement statistique en amont de l'étude (e.g. pour imputer des valeurs manquantes, ou correctement tenir compte des valeurs aberrantes). Pour autant, peu de travaux proposent de travailler sur des problèmes de modélisation en présence de données incomplètes.
Les données tabulaires et les séries temporelles n'ont pas reçu assez d'attention de la part de la communauté autour du deep learning, et nécessitent un traitement particulier. En effet, certaines métriques et fonctions de perte utilisées en analyse d'image ne sont pas applicables sur des données tabulaires. De même, vérifier qu'un modèle a bien estimé la distribution d'un ensemble d'images revient à vérifier "à la main" que le modèle génère des images (ou du texte) cohérentes, i.e. vérifier si le modèle dépasse les capacités humaines. Cela n'est en revanche pas applicable à un vecteur multivarié, et

nécessite des outils plus complexes pour comparer des distributions.
Cette thèse se concentre sur l'utilisation de méthodes de machine learning appliquées à des problèmes comportant des données incomplètes, dans un contexte assurantiel. Le secteur de l'assurance est en train de changer de paradigme avec d'une part la collecte de nouvelles données (e.g. les données GPS telematics, ou encore les photographies et les descriptions textuelles des sinistres), et d'autre part l'émergence de nouvelles méthodes d'analyse de données. Ce changement de paradigme représente un intérêt fort pour le secteur de l'assurance, car cela permet de mieux estimer le risque encouru par l'assureur de par son activité, et de mieux connaître ses clients.

Ce document regroupe trois contributions, regroupées dans trois chapitres distincts précédés d'un chapitre qui introduit les éléments techniques nécessaires à la compréhension des contributions. La première contribution propose une structure de modélisation permettant d'entraîner un modèle à prédire une variable cible fortement censurée. Nous appliquons notre méthode sur des problèmes de provisionnement en assurance, et comparons notre approche avec les outils standards de calcul de réserves en assurance, et montrons que nous donnons des prédictions plus précises en matière de variance.
La deuxième contribution propose une architecture permettant de combiner plusieurs jeux de données afin de construire un modèle prédictif sur chaque jeu de données qui tient compte de l'information prédictive présente dans chaque autre jeu. Les assureurs conçoivent souvent plusieurs produits différents couvrant le même risque. Cela est en partie causé par le fait que la population couverte est fondamentalement différente (e.g. pays différents, ou entités différentes de l'assurance). Cela conduit à des jeux de données distincts portant sur l'étude d'un risque commun, et les combiner permet potentiellement d'affiner l'étude du risque. Nous comparons notre approche proposée avec l'état de l'art, et montrons que nous donnons de meilleurs résultats.
La troisième contribution propose une architecture qui permet d'estimer la distribution conditionnelle d'une série temporelle. Cela permet à l'assureur de générer des scénarios économiques de type "monde réel", et d'anticiper des changements significatifs dans leurs placements financiers. Les méthodes usuelles employées pour répondre à ce type de problématique font en général une hypothèse structurelle sur le processus stochastique. Ici, nous proposons une méthode non-paramétrique pour estimer la distribution conditionnelle d'une série temporelle, afin d'en générer de nouvelles réalisations. Nous comparons notre approche avec l'état de l'art, et montrons qu'elle donne une meilleure estimation des auto-corrélations du processus stochastique, ainsi que de son support.

Chapter 1

Introduction

1.1 DAMI Chair of research

This thesis has been realized in the context of the Data Analytics and Models for Insurance[1] (DAMI) chair of research. The DAMI chair of research (chair of excellence) is the accomplishment of the relationship between BNP Paribas Cardif and the SAF laboratory[2] (Sciences Actuarielle et Financière), the latter being part of the ISFA school[3] (Institut de Sciences Financière et d'Assurances).
The DAMI chair of research is divided into two distinct research axes: the **models for insurance** branch consists of studying the influence of the regulatory and accountant environment on the construction of risk assessment models in insurance, whereas the **data analytics** branch focuses on the use of modern Machine Learning models to study the new challenges the insurer is facing with the multiple new available types of data (e.g. telematics data).
The work presented in this document is part of the data analytics research of the DAMI chair, and studies the use of machine learning methods to challenge existing methods in insurance.

[1] http://chaire-dami.fr/fr/
[2] https://isfa.univ-lyon1.fr/recherche/
[3] https://isfa.univ-lyon1.fr/

1.2 Importance of modern Machine Learning in insurance

It is of high interest for the insurance industry to foster the development of new statistical methods to improve their lines of business such as having more accurate loss reserves, a better asset and liability management or an appropriate pricing, among others. As an example, for regulation purposes, the insurer must have an accurate estimation of the underlying risk inherent to its business. A good knowledge of the risk allows the insurer to compute an appropriate premium for the customer, to calculate accurate loss reserves for the outcoming claims and to have a better estimation of its assets and liabilities, to name a few.
The insurance industry has a long history in econometrics research, carrying out studies with the state-of-the-art models as well as fostering partnerships with universities (e.g. the DAMI chair of research), which enable the insurer to have a profound knowledge in statistics and to improve its lines of business.

The deep learning community grew up sharply over the last two decades with the arrival of more efficient computational facilities (e.g. more performant GPUs, cloud computing, TPUs, etc.) and the emergence of big companies having their own research labs, carrying out state-of-the-art deep learning models and a skyrocketting research activity in the last 20 years.
However, such labs mainly work on a kind of data which is of high interest to the company hosting it, namely image data, video, text or raw audio. The insurance industry mainly manipulates standard tabular data (collecting information about the customer or a claim) or time series data. Tabular data and time series have not received enough attention from the deep learning community. Indeed, a lot of work focus on a kind of data which usually carries a lot of signal by nature (e.g. objects in an image, words in a text or in an audio file) but very little has been done to study the statistical properties of deep learning models applied to a multivariate random vector. The latter point motivates the works presented in this document, which are conducted in the context of the data analytics research axis of the DAMI chair.

This thesis contains works around two main axis: *incomplete data* and *statistical learning*. Statistical learning theory is a field at the confluence of computer science, statistics, functional analysis and signal processing, which focuses on the estimation of a function based on given observations. The estimation of such a function is usually

done by minimizing an appropriate loss function between a target variable and a function of the observed explanatory variables.
Statistical learning has experienced multiple paradigm shifts according to the increasing power of computational abilities.

Although statistical learning models are powerful tools to detect patterns and estimate distributions, a particular attention has to be paid when applying such algorithms to incomplete data. Indeed, most statistical learning methods are designed to work on a complete, clean dataset. In practice, having a dataset of good quality is challenging, almost no dataset is clean. Multiple phenomena can have a significant influence on the quality of statistical learning models, e.g. missing data, outliers, noisy data, censored and truncated data, among others.
In this work, we focus on the application of statistical learning models on incomplete data in an insurance context.

The work presented in this document is organized in three research papers, established in three distinct chapters, precedeed by an additional chapter introducing most of the technical elements that we use in each contribution.

Chapter 2 gives details about the different methods used throughout each contribution. We first dig through the multiple phenomena which lead to have incomplete data. We define censored data, which is encountered when modelling a time related random variable. We also discuss about missing data, which is a widely encountered phenomenon of incomplete data in practice. We then give a word about extreme values, which is a frequently observed phenomenon that can lead to a bias in the statistical analysis if it is not correctly considered. Finally we enlighten a particular incompleteness scheme, which is encountered when we want to carry out a statistical analysis based on multiple distinct datasets.

Chapter 3 constitutes the first contribution of this thesis. We study the use of statistical learning methods to compute the insurance loss reserves at an individual level. The problem of loss reserving is that the target variable is highly censored, hence the need to propose an appropriate framework to build an unbiased estimation of individual loss reserves. We compare our approach with the chain ladder, which is the standard tool in insurance to compute the loss reserves. Chain ladder requires strong assumptions on the structure of the censored target variable, and estimates loss

reserves at an aggregated level. We show that our approach outperforms the chain ladder on a real insurance dataset as well as on simulated examples.

Chapter 4 is the second contribution of this thesis. Modern machine learning methods usually require a large amount of data to be efficient. Such an amount of data may not be always available in a single dataset, which means that the information is stored in multiple distinct datasets. This context introduces what we call the structural incompleteness, which refers to the fact that two distinct datasets may not share the same attributes nor have the same distribution for common attributes. We propose a method that combines multiple datasets and addresses the problem of structural incompleteness by performing a style transfer through a latent representation of each dataset. We show that our approach outperforms the state-of-the-art method to leverage the predictive information from multiple datasets.

Chapter 5 is the third contribution of this thesis. In this chapter, we aim at estimating the distribution of a time series without making any assumption on the structure of the underlying stochastic process. For that matter, we propose an autoregressive deep generative model which estimates the conditional quantile function of the time series, while taking into account the fact that the stochastic process may be heavy tailed and high dimensional. Traditional generative models are designed to work on images (or on raw audio), which are low dimensional (usually 3 dimensions corresponding to the RGB colors), discrete and compactly supported data. We show that our approach outperforms the state-of-the-art model in terms of the estimation quality of the quantile function and of the inner properties of the time series.

Chapter 2

Background

This thesis focuses on utilizing modern machine learning methods on incomplete data, with applications to insurance. In the literature, incomplete data often refers to missing values, but in this work we extend the notion of incompleteness to an ensemble of phenomena which we define in this chapter. Most statistical learning methods are designed to work on a *complete* dataset, i.e. a sequence of *independent and identically distributed (i.i.d.)* random variables without any incompleteness phenomenon. In practice, almost no dataset is complete, and the data analyst often needs to perform a prior analysis on the data. Indeed, some incompleteness phenomena may have a significant influence on the statistical analysis, hence the need to ensure that such an analysis is not biased.
This chapter aims at introducing the technical elements used throughout the document. In this thesis, we will use the terms *statistical learning* and *machine learning* interchangeably. We introduce in Section 2.1 some incompleteness phenomena and discuss the impact of such schemes on the study. We then define the standard elements of statistical learning in Section 2.2 in order to enlighten the consequences that incompleteness schemes can have on the study. Chapters 4 and 5 make use of particular deep learning models. We define the technical elements of deep learning used in those chapters in Section 2.3. Chapter 5 deals with time series, which is a particular kind of data that needs an appropriate modelling. We define the usual properties and models for time series modelling in Section 2.4.

2.1 Incomplete data

Multiple phenomena can lead to data incompleteness. Some are completely random (e.g. a ponctual error of measure or a random loss of information), others are structural

and can depend on the data values or more complex patterns.
The underlying phenomena which lead to data incompleteness have been largely studied and some have a long history in the statistical literature. However, most methods handling incompleteness rely on strong structural assumptions, and literature on the use of statistical learning to address data incompleteness problems is still scarce.

This section gives an overview of some incompleteness phenomena and gives a word about existing work in each domain. The outline of this section is the following. Subsection 2.1.1 explains the censorship phenomenon, Subsection 2.1.2 focuses on the missing values and its associated missingness schemes, Subsection 2.1.3 introduces the study of rare events and Subsection 2.1.4 gives explanations on a particular missingness scheme.

2.1.1 Survival data

One form of data incompleteness occurs when studying a time-related phenomenon. Survival analysis focuses on the study of a random variable T, where T refers to a certain time or duration. The problem of studying such a random variable is that we usually have at our disposal a partial observation of the phenomenon. This is due to the fact that recording a duration requires to wait for the phenomenon to be terminated to have a full observation of T, whereas a "standard" record (i.e. not related to a duration) is usually measured and recorded instantly.

Recording observations of a random variable T usually resuires to record it over a certain time span $[t_0, t_1]$. Between time $t = t_0$ and time $t = t_1$, some observations of T will be fully observed, others will be partially observed, i.e. the outcome of the event will happen at a time $t > t_1$. Such a partial observation is *right censored*. Likewise, some occurences of T happened before $t = t_0$, and are said to be *left censored*. Figure 2.1 depicts the censorship phenomenon.

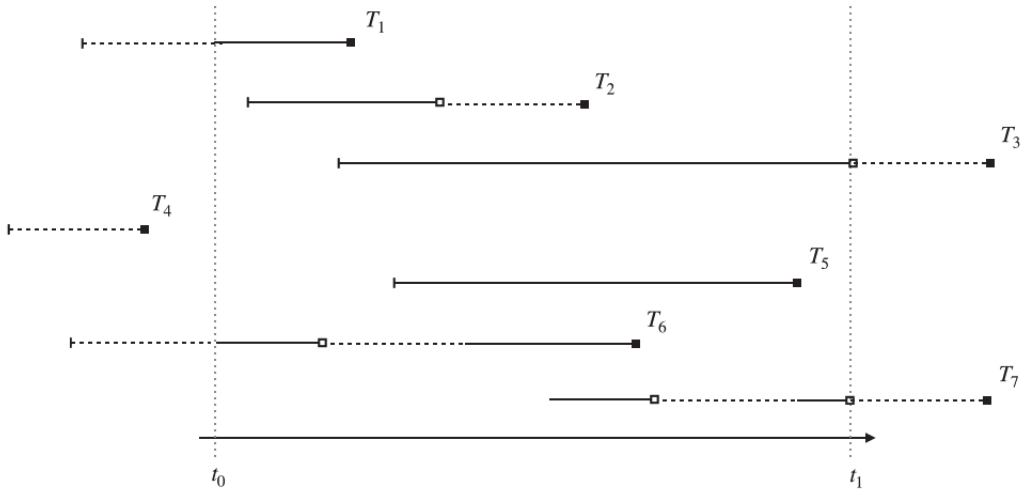

Fig. 2.1 Scheme describing the censorship phenomenon. Dashed lines (resp. plain lines) represent unobserved (resp. observed) values; plain squares represent the outcomes (e.g. death of an individual) and empty squares represent censored values (i.e. the latest known values without outcome). $T_1, \ldots, T_7$ are iid time-related random variables with distribution $p(T)$. We observe T in a timespan $[t_0, t_1]$. T_1, T_5 and T_6 are observed, T_2, T_3 and T_7 are right-censored, and T_4 is left censored.

Removing censored observations prior to a statistical analysis or considering that a censored value is a fully observed value introduces a bias in the analysis. Hence the need for appropriate statistical methods handling censored data.

Let $(T_1, X_1), \ldots, (T_n, X_n)$ be an iid sample drawn from a random vector (T, X), where $X \in \mathbb{R}^p$ is a vector of covariates which may have an influence on T. In practice, we do not directly observe (T_i, X_i) $(i \in \{1, \ldots, n\})$ but the vector (Y_i, δ_i, X_i) instead, where:

$$\begin{cases} Y_i &= \min(T_i, C_i), \\ \delta_i &= \mathbb{1}_{T_i < C_i}, \end{cases}$$

and $C_1, \ldots, C_n$ are iid copies of an unknown censoring random variable C such that $Y \perp\!\!\!\perp C$.

One main objective of survival analysis is to estimate the distribution of T. It is usually done by estimating distibution-related quantities such as the *survival function*

or the *hazard function*, among others. For a time $t > 0$, the survival function S of T is defined as the probability that a duration drawn from T goes beyond time t:

$$S(t) := \mathbb{P}(T > t) = 1 - F(t), \tag{2.1.1}$$

where F is the *cumulative distribution function (c.d.f.)* of T. In other words, when T represents a duration for an individual, e.g. life expectancy, $S(t)$ is the probability that an individual survives beyond time t.
The *probability density function (p.d.f.)* f of T is trivially related to the survival function by the following formula:

$$f(t) = -\frac{dS(t)}{dt}. \tag{2.1.2}$$

For a scalar random variable Z of c.d.f. F, the estimation of F using an iid sample $z_1, \ldots, z_n \sim Z$ is given by the empirical c.d.f. F_n:

$$F_n(z) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{z_i \leq z}, \quad z \in \mathbb{R}. \tag{2.1.3}$$

However, the estimation of the c.d.f. of a time-related random variable T cannot be done by using the formula of Equation 2.1.3 due to censored observations (which means that some z_j are not fully observed). Kaplan and Meier proposed a non-parametric estimator [Kaplan and Meier, 1958] to estimate the c.d.f. of such a random variable:

$$F_n^{\mathrm{KM}}(t) := 1 - \prod_{Y_i \leq t} \left(1 - \frac{\delta_i}{\sum_{j=1}^{n} \mathbb{1}_{Y_j \geq Y_i}}\right). \tag{2.1.4}$$

This estimator is still widely used in practice and is the baseline of many works of survival analysis. Another way to estimate the survival function is to compute the *cumulative hazard function* H which is defined as:

$$H(t) = \int_0^t h(x)dx, \tag{2.1.5}$$

where h is the *hazard function* which is defined by the following limit:

$$h(t) = \lim_{dt \to 0} \frac{\mathbb{P}(t < T < t + dt | T > t)}{dt}. \tag{2.1.6}$$

In other words, the hazard function is the rate of events occurring for individuals who survived beyond time t. In practice, we discretize the timespan $[t_0, t_1]$ and compute the values of h according to the discretization, then deduce the survival function from the following relation:

$$S(t) = \exp(-H(t)). \tag{2.1.7}$$

One other main objective of survival analysis is to predict a duration based on a set of explanatory variables. This enables the study of the influence of each explanatory variable on the duration. It is different from the previous objective since it does not estimate the distribution of T but the distribution of T given X instead. One of the most widely used model to perform a regression on a time-related random variable is the so-called Cox model [Cox, 1972] which models the hazard function by:

$$h(t; x) = h_0(t) \exp(\beta' x), \tag{2.1.8}$$

where x is a given vector of explanatory variables, β is a vector of parameters to estimate and h_0 is a *baseline hazard* which corresponds to the hazard when $x = 0$. Note that such a model assumes that individuals with distinct attributes have proportional hazard functions.

Litterature on how to estimate the survival function and how to perform a regression on a time-related random variable is rich and instigated a lot of interest in the medical field as well as in the insurance industry.
However, literature on how to perform a regression on a time variable by using modern statistical learning methods is still scarce. [Lopez et al., 2016] propose to use a Classification and Regression Tree (CART) [Breiman et al., 1984] to perform a regression on a time variable by weighting each terminal node by the Kaplan-Meier survival function, therefore taking into account the censorship. Some working papers such as [Gerber et al., 2018] carry on the same idea, and [Kuo, 2018] trains a neural network to estimate aggregated values of time-related random variables (loss reserving in insurance, at a macro level), but to the best of our knowledge, no other thorough work has been done on the regression task under censorship.

In Chapter 3, we propose to use a statistical learning algorithm - the ExtraTrees model [Geurts et al., 2006] - to accurately estimate individual loss reserves in insurance, and show that our approach outperforms the state-of-the-art loss reserving models.
Loss reserving refers to the computation of the amount of money that the insurer have to reserve to anticipate the outcoming future claims. From an individual point of view, when a policy holder declares a claim, reserving corresponds to the computation of the

total amount that the insurer will have to pay until the claim is settled, e.g. monthly payments of a loan (in case of unemployment or disability), legal actions (e.g. lawyers or fines) or car repairs, among others. The total amount of a claim is usually unknown at time of declaration, and more and more information about the final amount are collected over time. This means that the total loss amount for each claim is censored, which motivates the modelling of a claim amount with survival analysis methods.
Such a task is usually done by aggregating the losses of an insurance portfolio (the chain ladder), then the aggregated amounts are modelled with strong structural assumptions. Non-parametric literature to model such an aggregation is rare, and is even more rare when tackling the problem at an individual level, which motivated this chapter.

2.1.2 Missing values

Missing data is a pervasive issue in the everyday life of every data analyst. This is a well-known phenomenon that has a long history in the statistical literature, and many methods have been proposed to handle missing values, in many statistical settings. Almost all statistical methods are not designed to handle missing values, and generally expect a real-valued matrix as input. Hence, many works have been done to perform an appropriate data imputation prior to the statistical analysis.
This thesis does not present any contribution addressing the missing values issue with statistical learning. However, missing values clearly are one of the most encountered incompleteness scheme in practice, hence constitute a natural extension for future contributions, and a word has to be given about such a phenomenon.

Let $X = (X^{(1)}, \dots, X^{(d)})$ be a random variable taking its values in a d-dimensional real-valued space $\mathcal{X} = \mathcal{X}^{(1)} \times \dots \times \mathcal{X}^{(d)}$, and let $\mathcal{D} := \{x_i\}_{i=1}^n$ be a dataset of realizations of an iid sequence of random variables $X_1, \dots, X_n$ with distribution $p(X)$.
In practice we do not directly observe $\mathcal{D}$ but $\tilde{\mathcal{D}} := \{(\tilde{x_i}, m_i)\}_{i=1}^n$ instead, where $(m_1, \dots, m_n)$ are realizations of a sequence of random variables $M_1, \dots, M_n$ such that $M \in \{0,1\}^d$, and $(\tilde{x_1}, \dots, \tilde{x_n})$ are realizations of a sequence of random variables $\tilde{X}_1, \dots, \tilde{X}_n$ such that $\tilde{X}_i = (\tilde{X}_i^{(1)}, \dots, \tilde{X}_i^{(d)})$ defined as:

$$\tilde{X}_i^{(j)} = \begin{cases} X_i^{(j)} & \text{if } M_i^{(j)} = 1, \\ * & \text{otherwise,} \end{cases}$$

where $*$ is a missing value which is an out-of-support value for $\mathcal{X}^{(i)}$. In other words, M is the random variable representing the *mask* matrix indicating which components of X are observed. Note that we always observe M.

The theory behind the study of missing values was first developped by [Rubin, 1976], who defined three distinct mechanisms of missing values according to the dependency between M and X. As an example for illustrating the different mechanisms, suppose we make a census where the objective is to determine the average income of a population, and for some reason several income values are missing.
If M is independent of X ($M \perp\!\!\!\perp X$), then the missingness mechanism is said to be *Missing Completely At Random (MCAR)*. This scheme is the strongest assumption we can make on the missingness. It can be easily reproduced by considering a *complete* dataset (i.e. with no missing values) and randomly removing observations without taking into account the values of X. Such a missingness is usually caused by a random noise introduced while recording the data. In our example, this corresponds to the fact that by any chance some records have been lost between the collection and the analysis.
If M depends on X only through its observed values, then the missingness mechanism is said to be *Missing At Random (MAR)*. This is the most popular scheme in practice, and requires an appropriate imputation method which takes into account the dependencies between X and M. In our example, this corresponds to the fact that some sub-populations, say the top-managers (assuming we have this information at our disposal), were less willing to share their income. This means that among top managers, the missingness scheme is MCAR.
Otherwise if there are inner correlations between the components of M and dependencies between X and M, then the mechanism is said to be *Missing Not At Random (MNAR)*. In our example, this scheme corresponds to the fact that the missingness is caused by the values of the income itself. For example, say people who earns less than €1k/month are less likely share their income than others to the survey.

Strong theoretical and practical results are given in [Rubin, 1976] and [Little and Rubin, 2019] for estimating the distribution of X in presence of missing values. This is usually done by considering a parametric structure on the distribution of X and M (which depends on the assumed missingness mechanism) and estimating its parameters by using an Expectation-Maximization (EM) algorithm [Dempster et al., 1977]. Such approaches allow to estimate the distribution of X (based on $\tilde{X}$) and are still widely used in

practice to perform a statistical analysis in presence of missing values. However, this requires a strong assumption on the structure of X and M.

As mentioned above, most statistical models are not designed to directly handle missing values and often require a fully observed iid sequence $X_1, \dots, X_n \sim X$. Therefore, it is of high interest to perform an appropriate data imputation prior to the analysis. Among the many existing imputation methods, the simplest one is to impute the missing entries by the mean of the observed attributes. The latter is highly criticised in [Little and Rubin, 2019] since it highly distorts the distribution of X, but paradoxically this is one of the most employed method in practice (except in domains such as in medical settings or clinical studies, where the methodology used to impute values is highly controlled).
There exist many other ways to impute missing entries of a given dataset. MissForest [Stekhoven and Bühlmann, 2011] consists of training a Random Forest algorithm [Breiman, 2001] to predict the missing entries based on their other attributes. The latter approach is fairly used in practice because it is relatively simple to implement and can handle mixed types of data to be imputed (discrete and continuous). Note that the MissForest algorithm can handle MCAR and MAR mechanism.

Deep learning approaches have been proposed to impute data, such as [Vincent et al., 2008] who use an Auto-Encoder[1] (AE) model which takes a noisy input and is trained to output a denoised output (Denoiser Auto-Encoder, DAE). However, this method is trained by using a complete dataset of images in which the authors randomly change the color of some pixels (note that this can be adapted to tabular data by randomly removing some values of $\mathcal{D}$). This means that such a model i) needs a complete dataset to be trained, and ii) only consider the MCAR mechanism.

Other methods using Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] have been proposed. In particular, [Li et al., 2017] uses a GAN framework to denoise images representing faces. The model is trained by injecting noise in faces and training it to reconstruct the faces. This model gives excellent results for imputation, however i) it requires a complete dataset to be trained, and ii) face reconstruction is a very

[1] An Auto-Encoder (AE) is a model which is trained to output its input. The simplest AE model consists of training two Multi-Layer Perceptron (MLP) F, G, where $F : \mathcal{X} \to \mathcal{Z}$ *encodes* the data to a latent space $\mathcal{Z}$, and $G : \mathcal{Z} \to \mathcal{X}$ *decodes* back to the original feature space. F and G are joinly trained to minimize a reconstruction error, and if F and G have no hidden layer, then the model $H(\cdot) := G(F(\cdot))$ is an MLP with exactly one hidden layer.

specific task which can be difficult to generalize on tabular data.
On the other hand, [Yoon et al., 2018a] trains a GAN to impute plausible values for a dataset which contains missing values. The authors show that their approach outperforms the usual imputation methods with respect to a specific prediction task for several real-world datasets. However, such an approach does not give any insight on the imputation quality in terms of the preservation of the structure of X, and no discussion is given regarding the missingness mechanism. Hence comparing this method with MissForest (and other methods) may not be appropriate and relevant.

Multiple Imputation

Independently of the imputation method, [Rubin, 2004] showed that performing a single imputation on missing values (i.e. exactly one prediction of a missing entry) can lead to systematically underestimate the variance of the distribution of X, and proposed to address this issue by performing a *multiple imputation* on the dataset instead.
Multiple imputation refers to the fact that instead of predicting only one value for a missing entry, we rather predict multiple plausible values for that entry, which means that we generate multiple imputed datasets, then perform a statistical analysis on each imputed datasets and combine the results.

A widely used method to perform multiple imputation is the Multiple Imputation by Chained Equation (MICE) approach [van Buuren and Groothuis-Oudshoorn, 2011]. MICE usually consists of assuming a parametric structure on X, then sampling (with a Gibbs sampler) values along each dimension of X from the other dimensions (which are iteratively imputed thanks to the updated estimated parameters) and then update the parameters along with the generated samples.
From the side of the deep learning literature, [Gondara and Wang, 2017] propose an adaptation of the DAE [Vincent et al., 2008] for multiple imputation. To the best of our knowledge, no other thorough work have been done on multiple imputation on the deep learning literature.

2.1.3 Extreme values

Extreme Value Theory (EVT) is a field of probability and statistics which aims at modelling rare events. Rare events are particularly studied in the insurance industry since they can lead to large claim amounts. Natural disasters such as the earthquake and tsunami in Japan that occured in 2011 - which led to a total losses estimation of

€147 billions - are examples of rare events.

This thesis does not make any contribution on the EVT field nor use EVT results, however Chapter 5 focuses on the estimation of the conditional distribution of a time series, which can be heavy tailed, e.g. financial time series. Unlike image processing or raw audio modelling, a stochastic process takes its values in a non-compact, real-valued space, which requires an appropriate modelling method. We propose an architecture to accurately estimate the conditional distribution of a time series, which takes into account the fact that it may be heavy tailed. EVT gives powerful tools to model the distribution tails of a random variable, and we introduce the general concepts of rare events modelling in this subsection.

From the statistical point of view, a rare event is an event of which the probability of occuring is low. The difficulties in studying rare events are that:

- most of the observed data is concentrated around the mean of the distribution,
- there is a low amount of observed data which are located on the distribution tails,
- below $\min(x_1, \dots, x_n)$ and beyond $\max(x_1, \dots, x_n)$, there is no more observation of the distribution tail, and estimating the distribution tail needs to estimate unseen values which is a difficult task.

The usual methods for modelling rare events is to assume a particular structure along the distribution tail and estimate its parameters. The Fisher–Tippett–Gnedenko theorem stands that for an iid sequence of random variables $X_1, \dots, X_n$ and its associated maximum $M_n = \max(X_1, \dots, X_n)$, if there exists a sequence of pairs $(a_n, b_n) \in \mathbb{R}_+^* \times \mathbb{R}$ such that

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{M_n - b_n}{a_n} \leq x\right) = F(x),$$

then F belongs to the family of *Generalized Extreme Value (GEV)* distributions with parameter ξ of which the c.d.f. is of the following form:

$$F(s;\xi) = \begin{cases} \exp(-(1+\xi x)^{1/\xi}) & \text{if } \xi \neq 0, \\ \exp(-\exp(-x)) & \text{otherwise.} \end{cases}$$

The GEV distributions are widely used in practice to model the asymptotic properties of the distribution tail of a sequence of random variables.
In the first place, works on statistical learning dealing with extreme values mainly focused on anomaly detection [Denning, 1987, Anderson, 1980, Lunt, 1990, Kumar, 1995] and the use of modern machine learning to detect intrusions have attracted a lot of attention [Lane and Brodley, 1997, Shon and Moon, 2007] and many tools have been developped to address such a task [E. King, 2009]. More generally, works on extreme values and statistical learning are gathering more and more interest and strong results (either theoretical or practical) have been established in the last two decades. To cite a few, [Roos et al., 2006] determined an upper bound (which depends on the observed dataset) of the difference between the train error and the generalization error, which is useful when there are unseen extreme values; [Goix et al., 2016] propose an algorithm based on low-rank models to detect anomalies in a high dimensional dataset; [Brownlees et al., 2015] study the Empirical Risk Minimization (ERM) when the loss function is not bounded (and yet is willing to have large values); [Jalalzai et al., 2018] propose a general framework (with theoretical results) to address a binary classification task where the explanatory variables X contain extreme values (in such cases, traditional approaches usually perform poorly due to the rarity of observations in the extremes). Deep learning literature on extreme values is still rare, and the few existing works focus on intrusion detection [Xu et al., 2015, Javaid et al., 2016, Du et al., 2017], and does not combine the use of deep learning tools with extreme value theory, which constitutes a natural extension for future contributions.

2.1.4 Cross-Domain Incompleteness

Cross-Domain incompleteness refers to a particular missing values mechanism, usually of type MNAR, which arises when combining multiple datasets (multiple domains) in a single statistical study (cross-domain). Assume we have at our disposal an ensemble of M distinct datasets $\mathcal{D}_1, \ldots, \mathcal{D}_M$, each dataset recording values of the same nature (e.g. distinct censuses about income, datasets from multiple hospital recording the same disease on patients etc.).
Many statistical methods require a large amount of data to be efficient, therefore it is of interest to utilize $\mathcal{D}_1, \ldots, \mathcal{D}_M$ to conduct the same statistical analysis, in order to leverage the information lying in each dataset.

Combining multiple datasets raises two main issues: *feature mismatch* and *distribution mismatch*. Feature mismatch refers to the fact that two distinct datasets $\mathcal{D}_i$

and $\mathcal{D}_j$ may not share exactly the same attributes (e.g. $\mathcal{D}_i$ records the gender and $\mathcal{D}_j$ does not). Distribution mismatch refers to the fact that common attributes for two distinct datasets may not share the same distribution (e.g. the proportion of male and female may differ between $\mathcal{D}_i$ and $\mathcal{D}_j$).

The simplest way to combine multiple datasets of the same nature is to concatenate them. However, because of feature mismatch, attributes that are present in $\mathcal{D}_i$ but not in $\mathcal{D}_j$ will systematically introduce missing values in $\mathcal{D}_j$.
The imputation of such missing values cannot be done with the usual imputation tools such as MissForest [Stekhoven and Bühlmann, 2011]. Indeed, when training a Random Forest [Breiman, 2001], it is assumed that the test set (to be imputed) shares the same distribution as the train set, which is not the case because of the distribution mismatch.

Chapter 4 focuses on combining multiple datasets to train a prediction model on each dataset $\mathcal{D}_1, \ldots, \mathcal{D}_M$ by training an appropriate *style transfer model* through a latent space. The prediction task is common for each dataset, and we show that our method gives better performances than the concatenation benchmark and than the state-of-the-art model.
Our proposed method can also be employed to perform an imputation in the same setting, since it takes into account the distribution mismach between each dataset. However, we only restrict our study to the prediction task, and further extended work has to be done to study the ability of our approach to preserve the structure of the distribution of each dataset.

2.2 Statistical Learning

Statistical learning refers to a particular field of statistics which makes an extensive use of non parametric tools, and gathers many fields such as mathematics, computer science, pattern recognition and optimization, to cite a few.
It is used to address many problems such as predicting an outcome based on some *features* (attributes), detecting patterns in multivariate distributions, machine translation, anomaly detection, among many others.

Let X be a random variable taking its values in a p-dimensional space $\mathcal{X}$ (typically $\mathbb{R}^p$) and let Y be an outcome of interest taking its values in $\mathcal{Y}$. According to the

task, $\mathcal{Y}$ can be either discrete (e.g. binary classification) or continuous (e.g. regression of an income). In statistical learning, we distinguish two main focuses: *supervised learning* and *unsupervised learning*.
There is no formal definition of supervised and unsupervised learning, and methods from supervised learning can be used to address unsupervised tasks. Unsupervised learning usually refers to the task of extracting patterns from an observed distribution of a multivariate random vector X.
Supervised learning refers to the task of predicting an outcome Y based on an observed ensemble of explanatory variables (attributes) X. This is done by seeking a function $f : \mathcal{X} \to \mathcal{Y}$ for predicting Y given the values of X, assuming a certain model for Y. For example, the following model estimates the conditional expectation of Y given X:

$$Y = f(X) + \varepsilon, \tag{2.2.1}$$

where f is a function belonging to a certain class of functions $\mathcal{F}$, and ε is a noise term, independent of X, such that $\mathbb{E}[\varepsilon] = 0$, typically a *white noise* (which we define in Section 2.4).

The estimation of f is usually done by minimizing the expectation of a certain *loss function* $\mathcal{L}$ conditionally on the observed covariates X, which is called *the risk* in the statistical learning literature. Hence, the estimation of f can be defined by the following optimization problem:

$$f^* := \arg\min_{f \in \mathcal{F}} \mathbb{E}[\mathcal{L}(Y, f(X))], \tag{2.2.2}$$

where f^* is the theoretical optimum function with respect to the class $\mathcal{F}$, which is called *the oracle*. In practice, we only observe a finite number of observations $(x_1, y_1), \ldots, (x_n, y_n)$ of the couple (X, Y), and we optimize the empirical version of the risk of Equation 2.2.2:

$$\widehat{f} := \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i)). \tag{2.2.3}$$

The latter optimization problem is called the *Empirical Risk Minimization (ERM)*, and forms the base of the statistical learning theory.
In practice, we usually observe a dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$ on which we estimate the predictor function f based on Equation 2.2.3, and a dataset $\mathcal{D}_{\text{new}} := \{x_i\}_{i=1}^m$ on which the outcome has to be predicted. The predicted dataset is written $\widehat{\mathcal{D}_{\text{new}}} = \{(x_i, \widehat{f}(x_i))\}_{i=1}^m$.

The performance of a prediction model is usually measured by a metric criterion denoted $\mathcal{M}$. In practice, the data analyst will measure the performance of f by splitting $\mathcal{D}$ into a *train dataset* $\mathcal{D}_{\text{tr}}$ which is used to estimate the predictor function f, a *validation dataset* $\mathcal{D}_{\text{val}}$ which is used to tune the *hyper-parameters* of f (e.g. depth of a tree, learning rate, etc.) and a *test dataset* $\mathcal{D}_{\text{test}}$ on which we measure the performance of $\widehat{f}$ by using $\mathcal{M}$, such that:

$$\mathcal{D} = \mathcal{D}_{\text{tr}} \sqcup \mathcal{D}_{\text{val}} \sqcup \mathcal{D}_{\text{test}}, \tag{2.2.4}$$

where $\sqcup$ represents the disjoint union. For a predicted test dataset $\widehat{\mathcal{D}_{\text{test}}} = \{(x_i, \widehat{f}(x_i))\}_{i=1}^{n_{\text{test}}}$ and a metric $\mathcal{M}$, the performance of f is measured by $\mathcal{M}(\{y_i\}_{i=1}^{n_{\text{test}}}, \{f(x_i)\}_{i=1}^{n_{\text{test}}})$.

2.2.1 Linear Model

One of the oldest - and most popular - statistical learning model is the linear model. It has a long history in the statistical and econometric literature and has been largely studied. The class of functions $\mathcal{F}$ for such a model is the class of all linear functions of X, i.e.:

$$\mathcal{F} = \{X'\beta; \beta \in \mathbb{R}^p\}, \tag{2.2.5}$$

where $X' = (X_1, \ldots, X_p)$ is the input vector and β is the vector of parameters to be optimized. Hence, the underlying model of Equation 2.2.1 can then be written as:

$$Y = X'\beta + \varepsilon, \tag{2.2.6}$$

where ε is a noise term such that $\mathbb{E}[\varepsilon] = 0$ and $Var(\varepsilon) = \sigma^2$. The latter equation assumes the *homoscedasticity* of the model (i.e. the variance of the noise term is the same for each observation), but weaker assumptions are studied in the econometric literature.

The most widely used and convenient loss function (not only for the linear model) is the *squared error* loss, which is defined for a prediction $\widehat{Y}$ and a target outcome Y as the quadratical error:

$$\mathcal{L}(Y, \widehat{Y}) := (Y - \widehat{Y})^2. \tag{2.2.7}$$

Hence, the optimization problem of Equation 2.2.2 is then written:

$$f^* = \arg\min_f \mathbb{E}[(Y - f(X))^2], \tag{2.2.8}$$

and its empirical version for the linear model is therefore written as the following optimization problem:

$$\widehat{\beta} = \arg\min_\beta \frac{1}{n} \sum_i (y_i - x_i'\beta)^2, \tag{2.2.9}$$

where $x_i' = (x_i^1, \ldots, x_i^p)$. The solution of Equation 2.2.9 is given by the *Ordinary Least Squares (OLS)* estimator:

$$\beta^{\text{OLS}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}, \tag{2.2.10}$$

where $\mathbf{X}$ is an $n \times p$ matrix representing the input dataset, and $\mathbf{y}$ is an outcome vector of size n to be predicted. Note that β^{OLS} only exists if $\mathbf{X}'\mathbf{X}$ is non-singular.
The linear model can be written as a particular case of the empirical risk minimization of statistical learning, where the loss function $\mathcal{L}$ is the squared error loss, and the class of functions $\mathcal{F}$ is the class of all linear functions of X.

The choice of the loss function $\mathcal{L}$ is primordial when conducting a statistical learning analysis. Indeed, when $\mathcal{L}$ is the squared error, then the solution of Equation 2.2.2 is the conditional expectation:

$$\arg\min_f \mathbb{E}[(Y - f(X))^2] = \mathbb{E}[Y|X], \tag{2.2.11}$$

which means that

$$f^*(x) = \mathbb{E}[Y|X = x]. \tag{2.2.12}$$

In other words, the best predictor of Y at any point $X = x$ - with respect to the average squared error (L_2 loss) - is the conditional expectation. On the other hand, when $\mathcal{L}(Y, f(X)) = |Y - f(X)|$ (L_1 loss), then the solution of Equation 2.2.2 is the *median* of Y given X. Hence, the choice of an appropriate loss function is of high importance according to the task.
Quantile regression is a method used to estimate a specific point in a distribution. It is performed by optimizing the quantile loss function [Koenker and Hallock, 2001] which is defined as:

$$\rho_\tau(u) := u(\tau - \mathbb{1}_{\{u \leq 0\}}), \tag{2.2.13}$$

where $\tau \in [0, 1]$ is a target quantile and u is an estimation error which is the difference between the observation and the estimated quantile. Hence, by using the quantile loss function, it is possible to estimate the quantile function for multiple values of τ by training the model with the appropriate loss each time. However, the latter method needs to train a model for each quantile τ, which is computationally expensive. A more effective way is to estimate the whole quantile function of X, $F_X^{-1}(\tau)$, with a single model $f : [0, 1] \to \mathcal{X}$, which takes as argument a quantile τ and maps it to $F_X^{-1}(\tau)$. In practice, this is done by generating a uniform distribution $U \sim \mathcal{U}([0, 1])$ and solving the following problem:

$$\arg\min_f \mathbb{E}[\rho_\tau(X - f(U))]. \tag{2.2.14}$$

In Chapter 4, we perform a domain transfer between several multivariate vectors. We do so by training a style transfer model which optimizes a sum of two distinct losses, each corresponding to a constraint we give to our model. In Chapter 5, we train a neural network to estimate the conditional distribution of a stochastic process. We do so by estimating the conditional quantile function of the time series with an appropriate loss that we detail in the paper.

One other parameter of high importance is the choice of the class of functions $\mathcal{F}$. Indeed, the choice of a class function is determined by the algorithm used to perform the statistical analysis (e.g. linear model, gradient tree boosting, neural networks, etc.), each having a different *capacity*.
The capacity of a model refers to its ability to learn complex patterns in the training data, and is quantified by the Vapnik-Chervonenkis (VC) dimension [Vapnik, 1995]. Informally, the VC dimension is defined as the largest set of points that a binary classifier can *shatter* (i.e. find a perfect split of the data). For example, in dimension $d = 2$, a linear classifier can always shatter 3 points no matter how the positive and negative labels are distributed, but cannot always shatter 4 points. This means that the VC dimension of such a classifier is at least 3 and strictly less than 4, which means that it is exactly 3.

The VC dimension constitutes a central part of the statistical learning theory developped by Vapnik and Chervonenkis. The most important results in statistical

learning theory show that the discrepancy between the training error and the generalization error (i.e. test error) is upper-bounded by a quantity that grows along the model's capacity, but decreases as the number of observations grows.
In practice, it is a difficult task to estimate the VC dimension of a model, particularly for deep learning models where the type of architecture can sharply alter the VC dimension.

As the capacity of a statistical learning model grows, the variety of learnable functions gets wider and therefore the training error diminishes. We expect from a statistical model to i) have a low training error in order to estimate complex patterns in the dataset, and ii) have a small generalization error, in order to preserve the performances observed from the train to new unobserved values (i.e. $\mathcal{D}_{\text{test}}$). The two latter points respectively refer to *underfitting* and *overfitting*. Figure 2.2 illustrates the underfitting and overfitting phenomena.

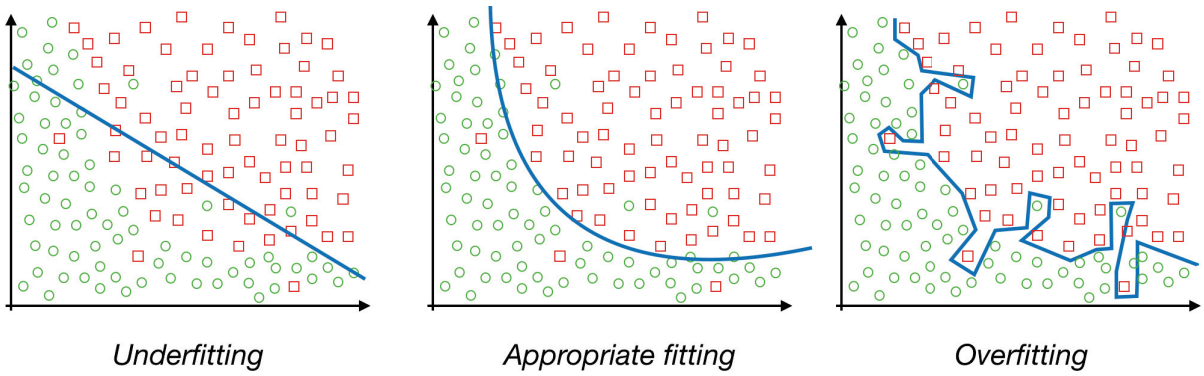

Fig. 2.2 Example of overfitting and underfitting. The left graph uses a linear model (i.e. low capacity) to fit the data, which is not appropriate since the underlying phenomenon is not linear. The right graph uses a model with too much capacity, and therefore will fail to generalize.

The whole stake of modelling with statistical learning tools is to find the optimal capacity of a model. This can be done by measuring the generalization error on a split of the training dataset as in Equation 2.2.4. A model with a too low capacity will struggle fitting the data, hence leading to a low variance but a high bias on the training set. On the other hand, a model with a too high capacity will learn "by heart" some patterns in $\mathcal{D}_{\text{train}}$ which will not generalize well in $\mathcal{D}_{\text{test}}$, leading to a low bias in the train set but a high variance. Figure 2.3 depicts the bias and variance tradeoff according to the model capacity.

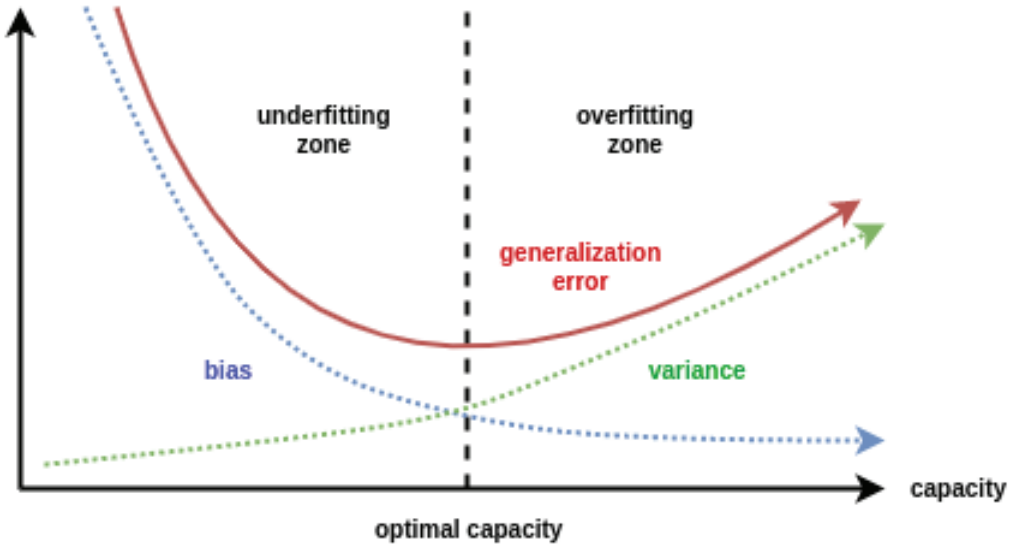

Fig. 2.3 Illustration of the bias and variance tradeoff according to the model capacity. A model with too low capacity will struggle fitting the data, hence inducing a low variance but a large bias. On the contrary, a model with too high capacity will learn signal that is basically noise and will not generalize, inducing a low bias in the train set but a large variance.

2.2.2 Bagging and Boosting

Bagging and boosting are two distinct methods for performing supervised learning, and are one of the most powerful and most widely used methods in practice.
Bootstrap aggregation (bagging) computes multiple prediction models on bootstraped replicas of a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$. More formally, the model to estimate the prediction function f is defined by the following formula:

$$f(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x), \tag{2.2.15}$$

where $B \in \mathbb{N}^*$ and f_b are prediction functions computed on b distinct bootstraped replicas $\mathcal{D}_1, \ldots, \mathcal{D}_b$ of $\mathcal{D}$.
One of the most powerful and widely used version of the bagging method is the Random Forest algorithm [Breiman, 2001], which uses a classification and regression tree (CART) [Breiman et al., 1984] as the prediction model f_b $\forall b$.

A regression tree consists of computing a partition of the input space $\mathcal{X}$ such that each

region has a distinct predictive power for the output Y. In other words, a classification tree f is defined as:

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{1}_{x \in R_m}, \tag{2.2.16}$$

where M is the number of distinct regions, $R_m \subset \mathcal{X}$ is the m-th region, and c_m is a constant value for region R_m.
Therefore, a tree is an estimated function constant by parts, where each constant c_m has a strong predictive power for Y. The tree is build by iteratively finding optimal splits of $\mathcal{X}$ to compute the regions $R_1, \ldots, R_m$. Regions are defined by a variable j on which the split is done, and a threshold s from which $\mathcal{X}$ is splitted. For example, R_1 and R_2 are defined as follow:

$$R_1(j, s) = \{\mathcal{X} | \mathcal{X}_j \leq s\}$$
$$R_2(j, s) = \{\mathcal{X} | \mathcal{X}_j > s\}.$$

The optimal splitting consists of computing j and s which are the solutions of the following minimization problem:

$$\arg\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} \mathcal{L}(y_i, c_1) + \min_{c_2} \sum_{x_i \in R_2(j,s)} \mathcal{L}(y_i, c_2) \right], \tag{2.2.17}$$

where $\mathcal{L}$ is an appropriate loss (objective) according to the prediction task. For example, if $\mathcal{L}$ is the quadratic loss, then Equation 2.2.17 can be written as:

$$\arg\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right], \tag{2.2.18}$$

and therefore the constants c_1, c_2 which solve the inner minimization problem are the average of y_i among observations belonging to R_1 and R_2 respectively, i.e.:

$$c_m = \frac{1}{card(\{x_i \in R_m\})} \sum_{x_i \in R_m} y_i. \tag{2.2.19}$$

The minimization problem of Equation 2.2.17 is also called in the literature the *impurity criterion*.

The tree is iteratively built by computing optimal regions $R_1, \ldots, R_m$ until reaching a stopping criterion which can be either a maximum number M of regions or a minimal number of observations in each region. In practice, the tree is grown until it reaches its stopping criterion, then it is *pruned*, i.e. some regions are gathered to satisfy a complexity criterion. More details about the growing procedure for classification and regression trees can be found in [Hastie et al., 2001].

Boosting refers to a specific family of models which consists of training en ensemble of M *weak* models, each model being trained from the errors of the previous models.
The predictor f is written as:

$$f(x) = \sum_{m=1}^{M} \alpha_m f_m(x), \tag{2.2.20}$$

where $\alpha_1, \ldots, \alpha_m$ are weights computed by the boosting algorithm, and reflect the respective contributions of each f_m for the prediction task. The boosting consists of iteratively building the weak predictors f_m, and giving weights $w_1, \ldots, w_n$ to the observations $(x_1, y_1), \ldots, (x_n, y_n)$, according to their misclassification distance (computed by the loss). For example, at the first step we set $w_1 = w_2 = \ldots = w_n = 1/n$, then train f_1 on $\mathcal{D}$, compute the loss for each observation and reweight each observation according to the error of f_1, in order to give more importance to misclassified observations for f_2.

Boosting is one of the most popular and powerful approach to perform supervised learning, as well on classification tasks as on regression tasks. The first version of this algorithm was first proposed by [Freund and Schapire, 1997] and the most popular version today are the XGBoost [Chen and Guestrin, 2016] and the LightGBM algorithm [Ke et al., 2017]. Such methods use the regression tree with low depth as the weak predictor function f_m.
Throughout this document, we focus on modelling complex phenomena by using statistical learning tools. In Chapter 3, we utilize an ExtraTree algorithm [Geurts et al., 2006], which is a variant of the Random Forest, to perform a regression in the case where the target variable Y is highly censored.

2.3 Neural Nets

Chapter 4 proposes a model for performing optimal transport between multiple multivariate domains while preserving the predictive signal of each domain, and Chapter

5 focuses on estimating the conditional distribution of a stochastic process. Both chapters use deep learning models, and this section aims at introducing the technical elements used throughout those chapters.

Neural networks are a class of statistical learning methods, used to address both supervised and unsupervised tasks, which constitute a keystone in modern machine learning modelling. Such models are not new and some have a long history in many distinct fields such as statistics, computer science and signal theory [Rosenblatt, 1958, Cybenko, 1989, Rumelhart et al., 1985, Rumelhart et al., 1988, Hinton et al., 2006, Bengio et al., 2007]. The popularity of neural networks skyrocketed in the last two decades, with the arrival of new methods performing faster optimization, coupled with the development of hardware tools allowing to train more complex models.

One of the most widely used families of architecture is the *multi-layer perceptron (MLP)*, also called *feedforward neural network* because the signal is propagated from an input x to intermediate *representations* until reaching the output y, with no backward connections. The signal is propagated through different *units* which can be of different nature (e.g. convolutions) according to the task to be addressed.

Let X be a random variable taking its values in a d-dimensional real-valued space denoted by $\mathcal{X}$ and Y be an output target variable taking its values in $\mathcal{Y}$ which can be either discrete or continuous. Neural networks are designed to satisfy the optimization problem of statistical learning presented in Equation 2.2.2, except that their nature allows to estimate a function f of high complexity. Therefore, the class of functions $\mathcal{F}$ belonging to neural networks is the class of all differentiable functions.

An MLP [Cybenko, 1989] with L *layers* is a function $f : \mathcal{X} \to \mathcal{Y}$ such that:

$$f(x) = f_1 \circ f_2 \circ \ldots \circ f_L(x), \tag{2.3.1}$$

where $\circ$ is the composition operator (i.e. $f \circ g(x) = f(g(x))$), and each layer $f_l : \mathbb{R}^{h_{i-1}} \to \mathbb{R}^{h_i}$ $(l = 1, \ldots, L)$ outputs a combination of its inputs:

$$(f_l(x))_i := w_0 + \sum_{j=1}^{d} w_j x_j \quad i \in \{1, \ldots h_i\}, \tag{2.3.2}$$

where $w_1, \ldots w_d$ are parameters which are computed to satisfy the optimization problem of Equation 2.2.2, and w_0 is a *bias* term (also computed to satisfy the minimization

problem). Note that the intermediate dimensions h_i of each layer are hyperparameters of an MLP, except for the input an output layers for which the dimensions are respectively d and $dim(\mathcal{Y})$. When there is one or more hidden layer in an MLP, then the number of vector parameters from Equation 2.3.2 increases and we rather consider the *weight matrix* to group the model parameters instead of separate vectors. In practice, we usually pass the output of a layer through an *activation function* in order to detect non-linearities in the intermediate representations of X. The most popular activation functions are the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.3.3}$$

the hyperbolic tangeant function

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}, \tag{2.3.4}$$

the Rectified Linear Unit (ReLU) $\text{ReLU}(x) = max(x, 0)$, the latter being very popular in practice, and the softmax function for a multidimensional output $x = (x_1, \ldots, x_K)$

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \quad i = 1, \ldots, K. \tag{2.3.5}$$

The softmax function is often used when $\mathcal{Y}$ is discrete since it allows to normalize the output such that it can be interpreted as a probability vector.

2.3.0.1 Optimization of a neural network

As for other statistical learning models, a neural network is trained to satisfy the optimization problem of Equation 2.2.2, where the function class $\mathcal{F}$ is the class of all differentiable functions of $\mathcal{X}$. The optimization is done by changing the values of the weights defined in Equation 2.3.2, and we denote by θ the parameter of the network containing its weights.

The optimization of a network is usually done by *backpropagation* [Rumelhart et al., 1985, Rumelhart et al., 1988], which consists of computing the gradient of f with respect to θ by iteratively computing its differentiate functions with the *chain rule*. The use of the chain rule is possible since a network can be written as a composition of multiple functions, i.e. its layers. The optimization of the parameters θ of the network is iteratively done by an appropriate optimization algorithm. One popular optimization

strategy is the *Stochastic Gradient Descent (SGD)* which is defined by the following recursion:

$$\theta^{t+1} = \theta^t - \lambda \nabla_{\theta^t} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f_{\theta^t}(x_i)), \tag{2.3.6}$$

where λ is a fixed learning rate, $\mathcal{L}$ is an appropriate loss function and θ^t are the weights at iteration t. Therefore, we need to compute the gradient in order to update the parameters θ^t. Although computing an analytical expression for the gradient in Equation 2.3.6 is relatively straightforward, evaluating it can be numerically expensive, which is the reason why we prefer to compute it by backpropagation.

The backpropagation algorithm lets the information from the loss flow backward through the network and compute the gradient by using the chain rule. Let h be the composition of two functions f, g such that $h(x) = f(g(x))$ and let $y = g(x)$, $z = f(y)$. The derivative of the function h is $h'(x) = f'(g(x))g'(x)$, which can be rewritten by using y and z with the Leibniz's notation:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}. \tag{2.3.7}$$

The above expression can be generalized in the case when $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}^n$:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}, \tag{2.3.8}$$

which can be written with matrix notations by using the Jacobian matrix $J_g(x)$ of g in x and the gradient of z in y:

$$\nabla_x z = J_g(x)' \nabla_y z. \tag{2.3.9}$$

The expression in the previous equation shows that computing the gradient of h is equivalent to computing the Jacobian matrix of g, then multiplying it by the gradient of h in y. Hence, this rule can also be applied when h is a composition of more than two functions, which is very effective since we compute the derivative of each function only once, the final gradient being a product.
Note that the expression of Equation 2.3.9 can aslo be generalized when $f \in \mathbb{R}^d$ instead of $\mathbb{R}$, and more generally to tensors of any dimension, but this requires complex notations which are not necessary to conceptually understand the chain rule and the backpropagation algorithm.

The SGD algorithm defined in Equation 2.3.6 is popular, but we often use an alternative version of this algorithm since the latter is quite slow in practice. [Sutskever et al., 2013] propose to incorporate the Nesterov's accelerated gradient method [Nesterov, 1983] as an alternative of the SGD, which takes into account the values of previous gradients in order to compute a velocity factor to accelerate the optimization process.
Also note that the learning rate is a crucial parameter in the optimization algorithm. In Equation 2.3.6, we set λ to be constant over time, but in practice it is better to consider a decreasing learning rate λ^t. Optimization strategies which adapts the learning rate have also been proposed. Among them, the most popular by far is the Adam algorithm [Kingma and Ba, 2014]. The RMSProp [Hinton, 2012] is also very popular, particularly when dealing with sequences and recurrent neural networks.

2.3.0.2 Recurrent neural networks

Recurrent neural networks (RNN) are a family of neural networks architectures which are designed to work with *sequences*. Sequences are observations for which the order has a significant importance, e.g. observations recorded over time or text data.
Such networks have many applications, mainly in text modelling where text data is considered as a discrete high dimensional random variable. RNNs have received a lot of attention from the deep learning community and have a wide range of applications to text data such as language modelling [Mikolov et al., 2010], speech recognition [Graves et al., 2013] or text generation [Graves, 2013], to cite a few. Such works on text modelling emerged with the recent advances in *word embedding* methods such as Word2Vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014] and FastText [Bojanowski et al., 2016, Joulin et al., 2016].

The particularity of RNNs is that each hidden unit takes as input i) the output of the previous cell and ii) an additional external input, corresponding to the next element in the input sequence. Each input and hidden state are given weights in order to modulate the importance of the past and current values of the sequence.
The standard framework of a recurrent neural network is depicted in Figure 2.4. Weights are associated to internal representations and to the input to dynamically moderate the signal according to the importance of past observations. A weight matrix U is associated to the input x, W is associated to the previous hidden state, and V is associated to the intermediate output of each cell. Hence, we can define the updating process with the following equations:

$$\begin{cases} i^{(t)} &= \theta_1 + Wh^{(t-1)} + Ux^{(t)}, \\ h^{(t)} &= \sigma(i^{(t)}), \\ o^{(t)} &= \theta_2 + Vh^{(t)}, \end{cases} \tag{2.3.10}$$

where i is the input state at time t, $h^{(t)}$ is the hidden state at time t and $o^{(t)}$ is the output state at time t, σ is an appropriate activation function (e.g. tanh), and θ_1, θ_2 are biases, which are jointly optimized with U, V and W.

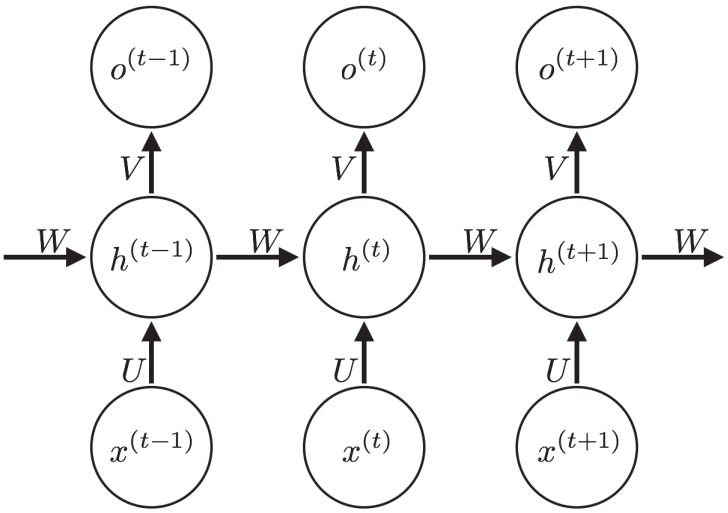

Fig. 2.4 Scheme of the standard framework of a Recurrent Neural Network (RNN). Each hidden unit takes as input i) the current observation $x^{(t)}$ of the sequence and ii) the previous hidden state $h^{(t-1)}$, and produce an output $o^{(t)}$ and an updated hidden state $h^{(t)}$.

2.3.0.3 Long short-term memory

Long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] is a family of RNN models which introduce a *self-loop* inside each unit and pass the signal through a *gating mechanism* which allows to dynamically change the time scale to use for past observations. This cannot be done natively with the standard scheme of an RNN.
The LSTM update procedure is defined by the following equations:

$$\begin{cases} f^{(t)} &= \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f), \\ i^{(t)} &= \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i), \\ o^{(t)} &= \sigma(W_o x^{(t)} + U_o h^{(t-1)} + b_o), \\ c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i_t \odot \tanh(W_c x^{(t)} + U_c h^{(t-1)} + b_c), \\ h^{(t)} &= o^{(t)} \odot \sigma(c^{(t)}), \end{cases} \tag{2.3.11}$$

where $\odot$ refers to the Hadamard (element-wise) product, $i^{(t)}$ (resp. $o^{(t)}$) is the input (resp. output) state at time t; $h^{(t)}$, $f^{(t)}$ and $c^{(t)}$ are respectively the hidden state, the forget gate and the cell state at time t. The forget gate f takes the same input as i except that it associates distinct weight matrices, and f aims at modulating the importance of previous cell states c in the updating of c. The update of the cell state is performed by a gating mechanism (i.e. associating distinct weight matrices to the inputs, and passing the signal through two distinct activation functions and aggregating along an element-wise product) associated to the forget gate. As for the standard RNN, the hidden state is updated by passing the input state through an activation function, except that it is multiplied element-wise by the cell-state (activated by an appropriate activation function). The components of an LSTM cell are depicted in Figure 2.5.

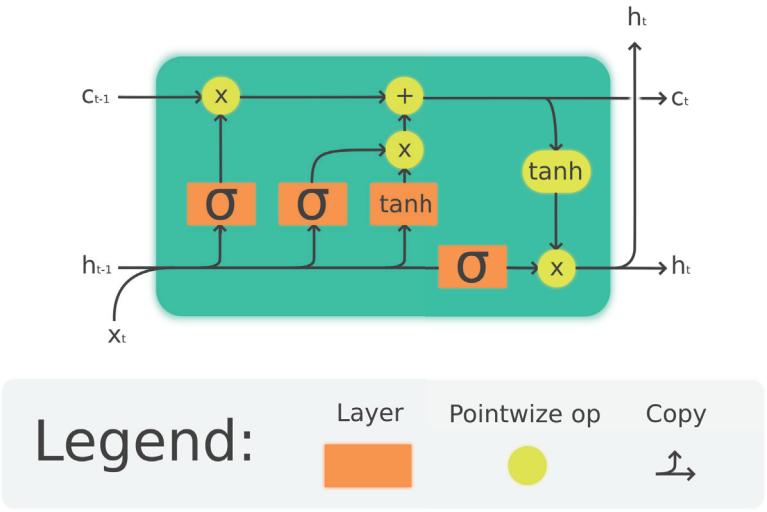

Fig. 2.5 Illustration of an LSTM cell. The inputs $x^{(t)}$ and $h^{(t-1)}$ are associated to four distinct weights matrices (orange blocks) with two distinct activation functions for the gating mechanism. The cell state is updated with the inputs and is then combined with the results of the gating in order to modulate the time scale to use for previous cell states.

The LSTM model was one of the first groundbreaking work for RNNs, and attracted many attention from the deep learning community. A popular variant of the LSTM is the Gated Recurrent Unit (GRU) [Cho et al., 2014] which uses a single gating unit to control both the forgetting factor and the hidden state. Although the state-of-the-art models on text analysis use more complex architecture (e.g. the BERT model [Devlin et al., 2018]), the LSTM and GRU models are still very popular and widely used in practice.

In this document, we do not make any contribution on the RNN field. However, Chapter 5 focuses on the estimation of the conditional distribution of a time series. A time series is a real-valued sequence vector of observations recorded over time and needs an appropriate modelling, hence the need of auto-regressive models. RNNs are natively auto-regressive, hence find natural applications in time series modelling [Connor et al., 1994, Giles et al., 2001, Esteban et al., 2017]. We propose in Chapter 5 an auto-regressive deep learning model which uses convolutions (CNN) to accurately estimate the conditional distribution of a time series, and compare our approach with the state-of-the-art models, which are RNN based.

2.3.0.4 Convolutional neural networks

Convolutional neural networks (CNN) are a family of neural networks which aims at modelling spatial dependencies of the observations through a convolutional operator. A convolutional layer consists of sliding a window (with associated weights to be optimized) through the data and perform a *convolution operation* inside the window. This allows to compute intermediate representations which can be aggregated by a *pooling layer*, in order to lower the dimension and detect strong patterns in the data. The idea of convolutional layers was first introduced by [LeCun et al., 1989] and convolutions are still widely used and very effective in practice.

For two real-valued functions f and g, the convolution operator is defined by the following integral:

$$(f * g)(t) := \int f(x)g(x-t)dx = \int f(t-x)g(x)dx. \tag{2.3.12}$$

In practice, f corresponds to the input data, and g refers to a *kernel* which corresponds to a weight matrix. The output of the convolution is referred in the literature as the

feature map. To compute the convolutional operator, we usually use the empirical version of Equation 2.3.12 where the support of x is discretized:

$$(f * g)(t) := \sum_{x=-\infty}^{+\infty} f(x)g(t-x) = \sum_{x=-\infty}^{+\infty} f(t-x)g(x). \tag{2.3.13}$$

In practice, we rather use the second expression of Equation 2.3.13 since it is more convenient to implement. Moreover, since the kernel size is fixed and usually lower than the dimension of the input x, the function g is then compactly supported, and the infinite sum in the previous equation only sums the elements inside the sliding kernel. The expression of the convolution operator in Equation 2.3.13 is defined for 1D data. However, one may have to work with higher dimensional data, hence the definition of the convolution operator can be extended to 2D data (e.g. images) by the following formula:

$$(f * g)(i, j) = \sum_m \sum_n f(i-m, j-n)g(m, n). \tag{2.3.14}$$

Most neural networks libraries such as TensorFlow[2] and PyTorch[3] implement a slightly different version of the convolution operator, the *cross-correlation* operator, which is defined for 1D data as follows:

$$(f * g)(t) = \sum_x f(t+x)g(x). \tag{2.3.15}$$

Figure 2.6 illustrates the convolutional operation on 2D data for a sliding kernel $g : \mathbb{R}^2 \to \mathbb{R}^2$, also called a 2×2 kernel in the deep learning literature. Once the feature map is computed, the signal is then passed through a non linear activation function and then passed through a *pooling layer*, which aims at reducing the dimension by aggregating each element in the feature map.

CNNs are a powerful tool to detect spatial correlations through the data, and are widely used in practice in most state-of-the-art architectures. Note that the weights of the kernels are shared thorough the data, which means that performing a convolution involve having less parameters to optimize, which results in faster training phases. In Chapter 5, we propose an architecture which makes use of convolutional layers (actually masked convolutional layers), in order to train a neural network to estimate the conditional distribution of a time series. We show that performing masked convolutions

[2]https://www.tensorflow.org/
[3]https://pytorch.org/

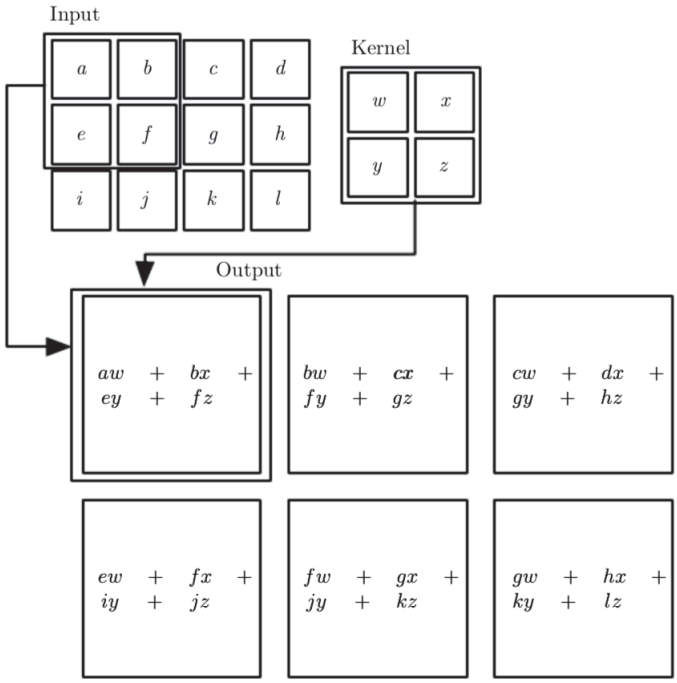

Fig. 2.6 Illustration of a 2×2 convolutional operation on 2D data. The kernel $g : \mathbb{R}^2 \to \mathbb{R}^2$ slides through the data and compute the cross-correlation between the data and the kernel, producing a new output at each step, called the feature map. Source image: [Goodfellow et al., 2016].

instead of training an RNN results in i) a faster convergence and ii) a more accurate estimation of the conditional distribution of the stochastic process. The proposed model belongs to a particular family of neural networks: the deep generative models, which we describe hereafter.

2.3.1 Generative modelling

Deep generative models are a family of neural networks which focus on the estimation of the distribution $\mathbb{P}_X$ of a multivariate random variable X without any prior assumption on the structure of $\mathbb{P}_X$. Such models allow to generate new samples which have the same distribution as X, and are used in many tasks such as image generation [Radford et al., 2015, van den Oord et al., 2016b, Pu et al., 2016], text generation [Yu et al., 2017, Guo et al., 2017] or raw audio generation [van den Oord et al., 2016a]. Such models are also used to detect inner components of $\mathbb{P}_X$, and allow to generate samples along specific attributes of the distribution, e.g. fader networks [Lample et al., 2017]

and multiview models [Chen and Denoyer, 2017].

There exist multiple families of generative networks, such as the Variational Auto-Encoders (VAE) [Kingma and Welling, 2013], Generative Adversarial Networks (GAN) [Goodfellow et al., 2014], Masked Autoencoders (MADE) [Germain et al., 2015] and PixelCNN [Oord et al., 2016, van den Oord et al., 2016b], among others.

VAEs are Auto-Encoders (AE) with an additional constraint on the latent representation of the data. An Auto-Encoder is a model which is trained to output its input, and is usually defined as a function h which is a composition of two functions f and g ($h(x) = g(f(x))$). For a random variable X which takes its values in $\mathcal{X}$, $f : \mathcal{X} \to \mathcal{Z}$ *encodes* the data into a latent space $\mathcal{Z}$, and $g : \mathcal{Z} \to \mathcal{X}$ *decodes* the latent representation back to the original feature space. f and g are jointly trained to minimize a loss function between X and $\widehat{X} := h(X) = g(f(X))$. VAEs set an additional loss on the latent representation $f(X)$, which is a Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951] between $f(X)$ and a Gaussian distribution $Z \sim \mathcal{N}(\mu, \sigma^2)$. In other words, VAEs constrain the latent representation to be Gaussian, which allows to generate new samples by generating $(z_1, \ldots, z_n) \sim \mathcal{N}(\mu, \sigma^2)$ and getting new samples of X with $(g(z_1), \ldots, g(z_n))$. Figure 2.7 depicts the block diagram of a VAE.
Note that in practice, we allow μ and σ to be multi-dimensional, which allows the model to detect mixtures in the distribution of X. VAEs are popular and easy to implement (a few lines of code) and gives interesting results. However, the constraint on the latent representation to be Gaussian indirectly forces the model to aggregate latent values, often resulting in blurry images when tackling images.

MADE are a family of feedforward networks, usually a Multi-Layer Perceptron (MLP) for which some weights from the weight matrix are forced to be zero. Zeroing such weights constrains the model to utilize only a part of the input dimensions, in order to output an estimate of the conditional distributions of all dimensions.
For a multivariate random variable $X \in \mathbb{R}^d$, the distribution $p(X)$ of X can be expressed by the product of its conditional univariate distributions with Bayesian arguments, i.e.:

$$p(X) = \prod_j p(X_j | X_{j-1}, X_{j-2}, \ldots, X_1). \tag{2.3.16}$$

The MADE model is an MLP trained to output its input (as for the Auto-Encoder) with a masked weight matrix. The choice of the mask is crucial and can be built

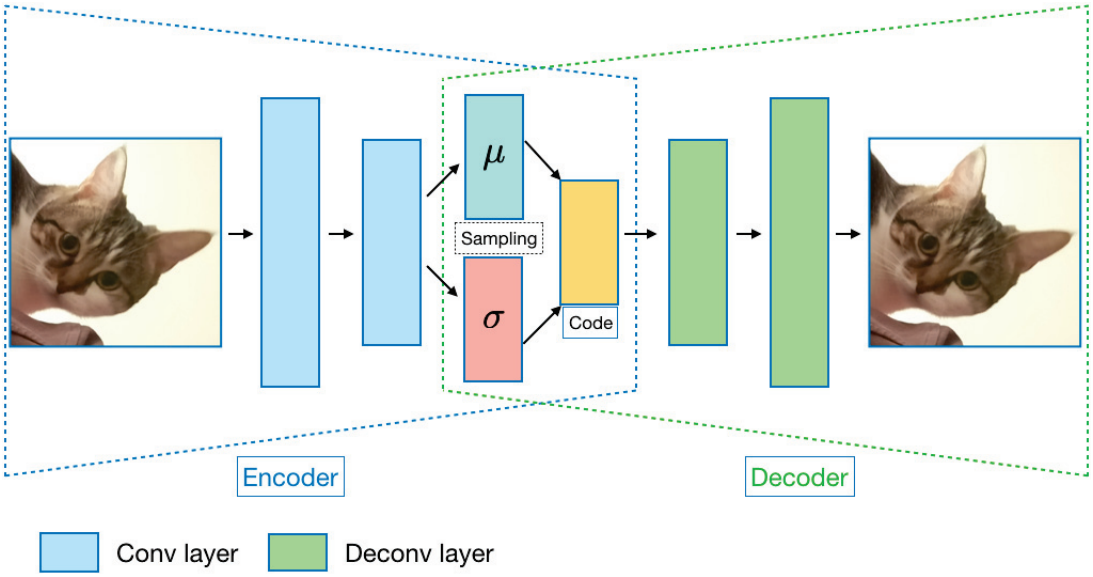

Fig. 2.7 Block diagram of a Variational Auto-Encoder. The model is trained to output its input via a composition of an *encoder* and a *decoder* and is given an additional constraint on its latent representation to be Gaussian distributed. To generate new samples, we generate Gaussian observations and decode it via the decoder to the original feature space and get the new image.

automatically. More details about this model can be found in [Germain et al., 2015]. Figure 2.8 illustrates the architecture of the MADE.

Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] are a family of models which consist of training two neural networks: a *generator* $G : \mathcal{Z} \to \mathcal{X}$ which takes a random noise as input (usually Gaussian) and aims at generating samples with the same distribution as X, and a *discriminator* $D : \mathcal{X} \to [0, 1]$ which aims at predicting if a sample has the same distribution as X or not. Thus, G aims at inducing D in error while D is trained to distinguish real samples from generated samples. Therefore, G and D are *adversarial* and the loss of both models is defined by the following minimax problem:

$$\arg\min_G \left[\sup_D \left[\mathbb{E}_X[\log(D(X))] + \mathbb{E}_Z[\log(1 - D(G(Z)))] \right] \right]. \tag{2.3.17}$$

Hence, the GAN framework is a two-agent zero-sum problem, the whole stake being to find a Nash equilibrium between the losses of G and D. Once the losses are balanced, G generates samples realistic enough so that D cannot decide whether the samples have the same distribution as X or not. Note that G can map any random noise

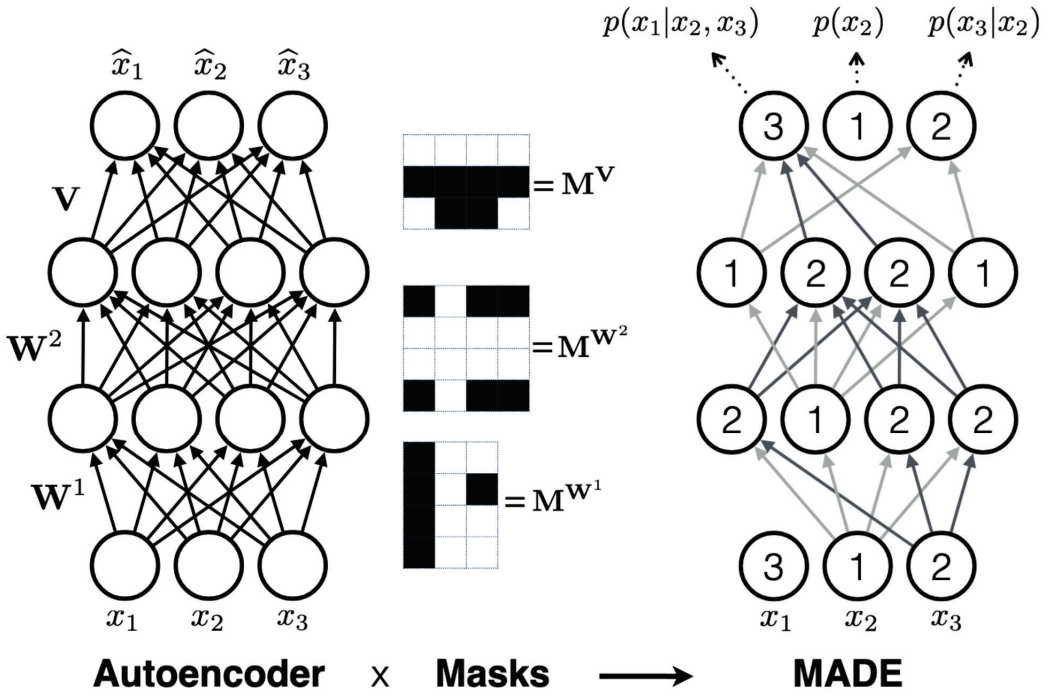

Fig. 2.8 Illustration of the MADE model. The model is an MLP which is trained to output its input. The associated weights matrix is masked in order to constrain the model to utilize a part of the input dimensions and to be autoregressive in its dimensions. Source image: [Germain et al., 2015]

independently of its initial distribution.
This method is a direct application of the optimal transport theory [Villani, 2009, Peyré et al., 2019] since the generator can transport a distribution to another distribution. Note that the VAE is also a direct application of optimal transport theory, but not the MADE, nor the PixelCNN of which we give details hereafter.

Finding an equilibrium in the losses of G and D is challenging as instabilities and issues come up in the training phase [Salimans et al., 2016]. As an example of such issues, we can cite the non-convergence of the minimax problem of Equation 2.3.17 or the *mode collapse* problem which refers to the fact that the generator only maps the random noise to one of the modes of the target distribution. An example of mode collapse in image processing is that G always generates one same image drawn from X (the image being real, D is then not able to distinguish if it is real or fake). This issue can be overcomed by giving D additional information about the sample to be predicted, such as inner similarities, which constrains G to generate diverse samples

(*minibatch discrimination*).
The issue of non-convergence have also been addressed with the Wasserstein GANs (WGAN) [Arjovsky et al., 2017] which consists of setting D to be real-supported, i.e. $D : \mathcal{X} \to \mathbb{R}$. This is done by removing the activation function of the last layer (usually a sigmoid) of D, and adding to the loss the norm of the gradient of D, which naturally contrains D to be Lipschitz and therefore ensures a better stability in the gradient computation.

Despite the troubles that the data analyst may encounter while training a GAN, this model is very popular in practice and has enabled most of the state-of-the-art results in many generative problems such as image processing [Radford et al., 2015, Isola et al., 2017, Ledig et al., 2017], text generation [Yu et al., 2017, Guo et al., 2017], image altering [Lample et al., 2017, Chen and Denoyer, 2017], domain transfer [Zhu et al., 2017a, Kim et al., 2017, Choi et al., 2018, Yoon et al., 2018b] or time series generation [Esteban et al., 2017, Zhang et al., 2018], to cite a few.

PixelCNN are a family of autoregressive neural networks which aim at estimating the conditional distribution of a multivariate random vector X. As for the MADE model, the PixelCNN utilizes the relation from Equation 2.3.16. The PixelCNN was originally proposed by [Oord et al., 2016] and decomposes the input shape to be vectorial. For example, an $n \times n$ image is decomposed to be an n^2 vector by concatenating all its rows into a single vector. Then, the model is trained to predict each pixel conditionally to the previous pixels. In the original paper, the authors used RNNs as the auto-regressive model (RNNs are natively autoregressive), however due to the long training time of RNNs, the authors later proposed to use CNNs with *masked convolutions* instead of RNNs.

Masked convolutions are defined by zeroing the weights associated to the pixels corresponding to future (unseen) pixels, and aim at making the model autoregressive. Figure 2.9 illustrates the masked convolution on 1D data (time series). The generation procedure for the PixelCNN is done iteratively by predicting the current pixel and feeding it back as input to the model to predict the next pixel. Figure 2.10 shows the iterative generation procedure for the PixelCNN.

In practice, PixelCNNs are quite popular, but not widely used because the generation procedure has to be iterative (generate values pixel by pixel) and takes a huge amount

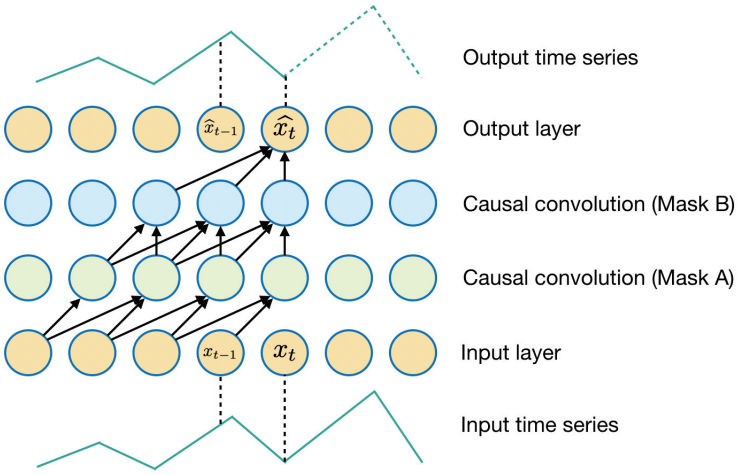

Fig. 2.9 Illustration of a masked convolution on 1D data. The weights associated to future values are forced to be zero so that the model is autoregressive.

of time when generating a large number of observations.
In Chapter 5 we propose a particular version of the PixelCNN which aims at estimating the conditional quantile function for a time series. PixelCNNs are usually performed on raw audio or on images, both having a discrete, low dimensional target, which is not the case for time series. We show that such an approach outperforms the state-of-the-art model which is a GAN model with an RNN network for G and D.

2.4 Time Series

Time series analysis refers to the study of a sequence vector of which observations are recorded over time. Examples of such vectors are financial time series (e.g. stock market), electricity consumption, electroencephalogram, among others.
Statistical analysis of time series has a long history in the econometric field, and strong modelling results have been established to address multiple problems related to time series such as analysis of the distribution, prediction, outlier detection, to cite a few.

A time series is a realization of a family of random variables $(X_t)_{t\in\mathbb{Z}}$, also called a *stochastic process*. A stochastic process $(X_t)_{t\in\mathbb{Z}}$ is said to be *strictly stationary* if the distribution of $(X_{t_1}, \dots, X_{t_n})$ is equal to the distribution of $(X_{t_1+h}, \dots, X_{t_n+h})$ $\forall n$, $\forall h$. In other words, a stochastic process is strictly stationary if all its sub-vectors are equidistributed, i.e. the distribution of $(X_t)_{t\in\mathbb{Z}}$ is time-invariant. In practice, besides

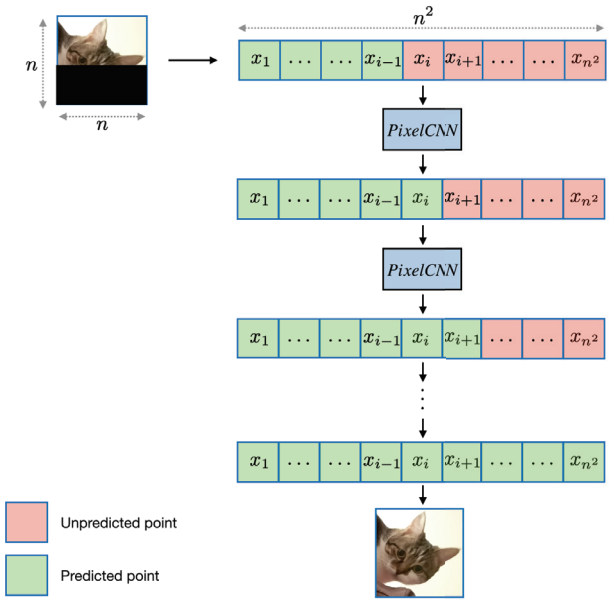

Fig. 2.10 Illustration of the generation procedure for PixelCNN. An $n \times n$ image is reashaped to an n^2 vector, and each predicted pixel is fed back as input for predicting the next pixel.

being a strong assumption, verifying the strict stationarity of a stochastic process can be difficult since it needs to estimate the distribution of $(X_t)_{t\in\mathbb{Z}}$.

For a stochastic process $(X_t)_{t\in\mathbb{Z}}$ of mean $\mu_t := \mathbb{E}[X_t]$, we say that $(X_t)_{t\in\mathbb{Z}}$ is *weakly stationary* if:

$$\begin{cases} \mu_t = \mu_{t+h} \quad \forall t, \forall h, \\ C_{XX}(t,s) = C_{XX}(t+h, s+h) \quad \forall t, s \; \forall h, \end{cases} \tag{2.4.1}$$

where $C_{XX}(t,s) = cov(X_t, X_s) = \mathbb{E}[(X_t - \mu_t)(X_s - \mu_s)]$ is the *autocovariance function (ACF)*. Hence, $(X_t)_{t\in\mathbb{Z}}$ is weakly stationary if its first moment and its ACF are time-invariant. In practice, it is more convenient to assume and verify the weak stationarity than the strict stationarity.

Structural models for time series usually assume a parametric structure on the stochastic process. Such a structure allows one to generate new samples of $(X_t)_{t\in\mathbb{Z}}$, as well as predicting future values of the time series. The core structure of time series modelling is to consider an auto-regressive model, i.e. a model where X_t depends on $(X_{t-1}, X_{t-2}, ...)$

with a noise term, and to assume a structure for the noise term.
The most convenient and widely used noise term in practice is the *strong white noise*, which is a series of iid random variables $(\varepsilon_t)_{t\in\mathbb{Z}}$ with the following characteristics:

$$\begin{cases} \mathbb{E}[\varepsilon_t] = 0, \\ Var(\varepsilon_t) = \sigma^2 < \infty. \end{cases} \tag{2.4.2}$$

Another popular noise term is the *weak white noise*, which is a series of random variables $(\varepsilon_t)_{t\in\mathbb{Z}}$ with the same characteristics as the strong white noise, except that the random variables are not independent but only need to satisfy $cov(\varepsilon_t, \varepsilon_s) = 0$, $t \neq s$. Throughout this document, unless explicitly precised, we will use the term *white noise* to design a *weak* white noise.

As mentioned above, the usual time series modelling methods assume a parametric structure on $(X_t)_{t\in\mathbb{Z}}$. One of the most popular and widely used linear structural model for time series modelling is the Auto-Regressive Moving Average (ARMA) model of order p and q, denoted ARMA(p, q), defined by the following recursion:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t, \tag{2.4.3}$$

where $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a white noise, and $\varphi_1, \ldots, \varphi_p, \theta_1, \ldots, \theta_q$ are the parameters of the model. An ARMA$(p, 0)$ is called an Auto-Regressive model of order p (AR(p)) and an ARMA$(0, q)$ is called a Moving Average model of order q (MA(q)). We respectively define the latter models based on the recursion from Equation 2.4.3:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t, \tag{2.4.4}$$

$$X_t = \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t. \tag{2.4.5}$$

Another popular structural model for time series is the Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) model [Bollerslev, 1986], which is an extension of the ARCH model [Engle, 1982]. Such a model is generally used to model the conditional variance σ_t^2 of a stochastic process. It is particularly used in financial time series modelling. A GARCH model of order p and q is defined by the following recursion on σ_t^2 and the squared values of $(X_t)_{t\in\mathbb{Z}}$:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i X_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2,$$

where α_i and β_j are the parameters of the model to be estimated, $X_t = \sigma_t \varepsilon_t$, and $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a white noise.

In practice, $(X_t)_{t\in\mathbb{Z}}$ is usually of dimension $d > 1$. Therefore, we model $(X_t)_{t\in\mathbb{Z}}$ with an appropriate model for multivariate time series. A popular approach to model $(X_t)_{t\in\mathbb{Z}}$ in higher dimension is the Vector Auto-Regressive model of order p (VAR(p)), which models each dimension of $(X_t)_{t\in\mathbb{Z}}$ with the following recursion:

$$\mathbf{X}_t = A^1 \mathbf{X}_{t-1} + A^2 \mathbf{X}_{t-2} + \cdots + A^p \mathbf{X}_{t-p} + \epsilon_t, \tag{2.4.6}$$

where the parameters to estimate are the matrices $A^k = (a_{i,j}^k)_{i,j\in\{1,\ldots,d\}}$, $k \in \{1, \ldots, p\}$, $\mathbf{X}_t := (X_t^{(1)}, \ldots, X_t^{(d)})'$ is the column vector of $(X_t)_{t\in\mathbb{Z}}$, and $\epsilon_t := (\varepsilon_t^{(1)}, \ldots, \varepsilon_t^{(d)})'$ is a vector of white noises.
The Dynamic Conditional Correlation (DCC-) GARCH model is a multivariate version of the GARCH model, and is defined as:

$$X_t = H_t^{1/2} \varepsilon_t,$$

where $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a vector of white noises, $H_t = D_t R_t D_t$, D_t is the diagonal matrix of conditional standard deviations of X_t, and R_t is the conditional correlation matrix for X_t.

Structural models have a long history in the time series literature, and many models have been proposed and strong results have been established [Box and Jenkins, 1970, Brockwell and Davis, 1991, Hamilton, 1994]. Works on non-parametric modelling for time series propose to model the time series as a function of its past values:

$$X_t = f(X_{t-1}, \ldots, X_{t-p}) + \varepsilon_t, \ \forall t \tag{2.4.7}$$

where ε_t is a white noise. However, the model defined in Equation 2.4.7 assumes that the variance of $(X_t)_{t\in\mathbb{Z}}$ is stationary. A more general expression of non-parametric modelling for $(X_t)_{t\in\mathbb{Z}}$ is to consider that the variance is also a function of the past values of $(X_t)_{t\in\mathbb{Z}}$:

$$X_t = f(X_{t-1}, \ldots, X_{t-p}) + g(X_{t-1}, \ldots, X_{t-p})\varepsilon_t. \ \forall t \tag{2.4.8}$$

The latter model allows to model a time series with a non-linear model, and many works have been done to propose a proper estimation of the functions f and g. [Robinson, 1983, Tsybakov, 1986, Hastie and Tibshirani, 1986].
However, deep learning literature on time series analysis is still scarce, and most of the work mainly focuses on time series prediction [Che et al., 2018, Binkowski et al., 2017, Borovykh et al., 2017]. Time series prediction refers to the estimation of the conditional expectations of its future values

$$\mathbb{E}[X_{t+h}|X_t, X_{t-1}, \ldots], \ \forall h > 0, \tag{2.4.9}$$

which is different from estimating the conditional distribution of $(X_t)_{t\in\mathbb{Z}}$, which is done by estimating its conditional univariate distributions since the following relation stands for any stochastic process:

$$p(X_{t+h}, \ldots, X_{t+1}|X_t, X_{t-1}, \ldots) = \prod_{j=1}^{h} p(X_{t+j}|X_{t+j-1}, \ldots). \tag{2.4.10}$$

Literature on deep generative modelling for time series is even scarcer, and the few works estimate the distribution by using a GAN framework with recurrent neural networks as the generator G and discriminator D [Esteban et al., 2017, Zhang et al., 2018].
As mentioned in the previous section, Chapter 5 focuses on the estimation of the conditional multivariate distribution of $(X_t)_{t\in\mathbb{Z}}$ with deep generative models.

Chapter 3

A Machine Learning approach for individual claim reserving in insurance

3.1 Introduction

With increased requirement on financial reporting and ongoing solvency concerns, actuaries are faced with the need, more than ever, to deliver reliable estimates of claim costs and reserves. A number of deterministic or stochastic methods based on aggregate claims data structured in claims development triangles exist for calculating outstanding claims reserves (e.g. Chain Ladder (CL) method, Bornhuetter-Ferguson method, ...). These methods have had a great success to manage reserve risk for a variety of lines of business but it is known that they can suffer from underlying strong assumptions that give rise to several issues. Among them, one can cite: i) an over parameterization risk induced by a large number of tail factors that have to be estimated as compared to the number of components of the run-off triangles, ii) a risk of error propagation through the development factors associated to a possible huge estimation error for the latest development periods, iii) a potential lack of robustness and the need for appropriate treatments of outliers, iv) the impossibility to separate the assessments of Incurred But Not Reported (IBNR) and Reported But Not Settled (RBNS) claims,...

The existence of these issues and the substantial literature about them indicates that the use of aggregate data is not so fully adequate for capturing the complexities of stochastic reserving for general insurance, mainly because of an important loss of information when aggregating the original individual claim data details (e.g. times of

occurrence, reporting delays, times and amounts of payments,...). Further, significant changes in technology (data collection, storage and analysis techniques) may make new approaches like a proper individual claims modeling now accessible.

Therefore, recent research strongly promotes claims reserving on individual claims data, see, for instance, [Antonio and Plat, 2014], [Badescu et al., 2016], [Hiabu et al., 2016], [Larsen, 2007], [Pigeon et al., 2013], [Taylor et al., 2008], [Verbelen et al., 2017], [Jin, 2013], among others... All these contributions assume a fixed and parametric structural form for which the distribution hypotheses have to be discussed and tested (e.g.[Pigeon et al., 2013] assumes a multivariate skew normal distribution to the claims payments). Such fixed structural forms are moreover not very flexible and are sometimes very difficult to estimate due to complex likelihood functions. Moreover the consideration of detailed feature information with a great data diversity is not always compatible with these rigid approaches.

On this basis, it has become crucial to implement more flexible models. Nowadays, Machine Learning techniques are very popular in data analytics and offer highly configurable and accurate algorithms that can deal with any sort of structured and unstructured information. [Wüthrich, 2018] has proposed for the first time a contribution to illustrate how the regression tree techniques can be used for individual claims reserving. However, for pedagogical purposes, he only considered the numbers of payments but not the claims amounts paid. Moreover he assumed that the claims occurrences and reporting process can be described by a homogeneous marked Poisson point process, and, as a consequence these numbers of incurred but not reported (IBNR) claims have been predicted by a Chain Ladder method exploiting the homogeneity assumption.

On this basis, we have decided to propose a new non-parametric and flexible approach to estimate separately individual IBNR and RBNS claims reserves that can: i) include the key claim characteristics in order to allow for claims heterogeneity and to take advantage of additional large datasets, ii) capture the specific development pattern of claims, including their occurrence, reporting and cash-flow features, and iii) detect potential trend changes, taking into account possible changes in the product mix, the legal context or the claims processing over time, to avoid potential biases in estimation and forecasting.

Our model is estimated on simulated data and the prediction results are compared with those generated by the Chain Ladder model. When evaluating the performance of our approach, we put emphasis on the impact of using micro-level information on the bias and variances of the prediction errors. Simulating data is indeed the only way

to have a precise idea of the performance of our new approach. Real data sets are in general short duration time series and the number of years (or quarters) to assess the bias and variances of the estimators is very small and can not be used to fairly compare algorithms. We therefore illustrate our method on a synthetic insurance portfolio but whose characteristics are very close to a real portfolio (we consider a (almost real) mobile phone insurance that covers the devices in the event of theft, breakage or oxidation). Simulating data gives us the opportunity to have at our disposal a very large number of independent computations of IBNR and RBNS claims reserves and then to derive precise estimates of bias and variances, which is definitively not possible with real data.

It is also important to underline that we decided to compare our non-parametric method with another non-parametric method, the Chain Ladder model, but not with some parametric individual claims reserving approaches to avoid discussions on the choices of their appropriate assumptions, which is against the objectives of our paper. It is true that the Chain Ladder model only uses aggregate claims data with no covariates, and it is expected that our approach will perform better. However we want to understand what the real benefits are in terms of bias and variances of the estimators and also to discuss the ability of our approach to still provide accurate estimators of the reserves when some of the assumptions of the Chain Ladder model do not hold.

We implement our new approach with an ExtraTrees algorithm but many other powerful machine learning algorithms can easily be adapted (RandomForest, XGBoost,...).

The outline of the paper is as follows. In Section 2 we introduce the claims reserving problem and we theoretically define the several types of outstanding claims reserves that we would like to estimate. In Section 3 we explain how we build the train and test sets on which the Machine Learning algorithms will be respectively trained and used to evaluate the individual outstanding claims reserves. Section 4 reminds the Chain Ladder methodology and gives methodological comparisons with our approach. In Section 5, we provide an explicit numerical example based on simulated data. We first describe the simulation scheme, then we list the individual claims covariates which are used by the ExtraTrees algorithm. We finally present and discuss the numerical results, and we compare them with those obtained by the Chain Ladder approach. In Section 6 we provide a short real case study based on a Dutch loan insurance portfolio. In Section 7 we give an outlook, provide a discussion and conclude.

3.2 The model

We consider an insurer who has been running business in one line of insurance and who is subject to claims reserves computation at some time t. Only outstanding claims for which liability has been assumed prior to time t are therefore relevant for this computation.

We start from a time-continuous setting where the policies' histories from their underwriting (including in particular the development of the claim if one occurs) are generated by a Position Dependent Marked Point Process (PDMPP). We equip the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ which satisfies the usual conditions. Note that such a PDMPP process is a mathematical model that can be used to model very general real data sets, and that no structural assumptions are made here, except for the measurability condition with respect to the filtration $\mathbb{F}$ (see e.g. [Arjas, 1989], [Norberg, 1993, Norberg, 1999] and [Haastrup and Arjas, 1996] for applications of such processes to model loss reserves with additional assumptions).

We assume that the insurer can use external information (if it is relevant) to predict the total outstanding payments, like e.g. the economic environment (unemployment rate, inflation rate, financial distress,...), or weather conditions and natural catastrophes (storm, flood, earthquake, etc.), and so on. We denote by $(E_t)_{t\geq 0}$ the (multivariate) continuous-time process that characterizes this information.

We now associate to each policy the following quantities:

- T_0: the underwriting date (Δ will denote the insured period and the contract will expire at $T_0 + \Delta$). Some features/risk factors are observed at T_0 by the insurer and can evolve over time: they will be plugged in the (multivariate) continuous-time process $(F_t)_{t\geq T_0}$. For example, for a life insurance policy, such features could be applicant's current age, applicant's gender (if allowed), height and weight of the applicant, health history, applicant's marital status, applicant's children, if any..., applicant's occupation, applicant's income, applicant's smoking habits or tobacco use)...

- T_1: the occurrence date of the claim ($T_1 = \infty$ if there is no claim). Only one claim is considered during the insured period in this paper for expository purpose (but the algorithm implemented in our computer program can deal with any number of claims).

- T_2: the reporting date (T_2 is assumed to be infinite by the insurance company until the claim is reported).

- T_3: the settlement date.

  During the settlement period the insurance company receives information on the individual claim like the exact cause of accident, the type of accident, the claims assessment and the predictions by claims adjusters (incurred losses), the external expertise, etc. We denote by $(I_t)_{t\geq T_2}$ the (multivariate) continuous-time process that characterizes this information.

  The settlement period is the period within transitional claim payments are done. We denote by $(P_t)_{T_2<t\leq T_3}$ the multivariate cumulated payment process (possibly negative). We let $P_t = 0$ for $T_1 < t \leq T_2$. The payments could be broken down into several components in case of several insurance coverages and if some legal and claims expert fees must be paid.

The mark associated to a policy is therefore given by

$$\mathcal{Z} = \left\{(F_t)_{t\geq T_0}, T_1, T_2, T_3, P_{T_1<t\leq T_3,}, I_{T_2<t\leq T_3}\right\}.$$

The insurer's portfolio is characterized by the collection of points $(T_{0,p}, \mathcal{Z}_p)_{p\geq 1}$ where the $\mathcal{Z}_p$ are in the space of policies' marks, as represented in Figure 1.

Note that we do not assume that the points $(T_{0,p}, \mathcal{Z}_p)_{p\geq 1}$ are independent. In particular common inflation effects can be taken into account in a component of the multivariate continuous-time process $(E_t)_{t\geq 0}$ (that is common to every policies) and create an hidden dependence between policies.

3.2.1 Categories of outstanding claims

First it should be underlined that T_1 must be smaller than $T_0 + \Delta$ for the insurance company to be liable for the claim because the contract must not be terminated at the claim occurrence. Let t denote the reserving date. If $t < T_1 < (T_0 + \Delta)$, the (potential) claim has not yet occurred at date t and therefore it is not considered as an outstanding claim.

Let $a \wedge b$ denote the minimum between the two constants a and b. The claims (that have occurred before t) may be categorized into two categories.

- If $T_1 < t < T_2$, the claim has occurred but it has not yet been reported to the insurance company. This claim is called an Incurred But Not Reported (IBNR) claim. For such a claim we do not have individual claim specific information,

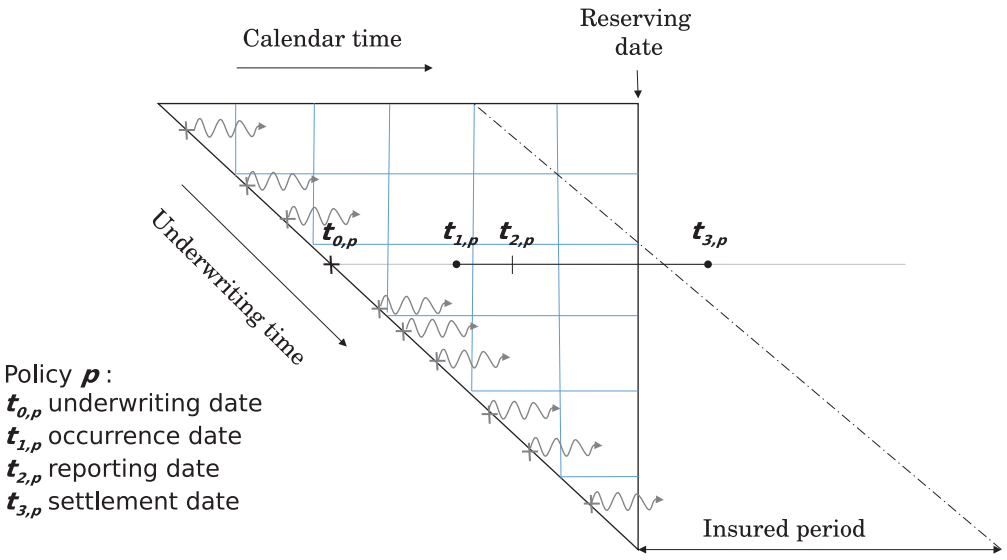

Fig. 3.1 Graphical representation of the position-dependent marked point process

but we can use the risk factors as well as external information to predict the likelihood of its occurrence and its cumulated payment. We hence define the $IBNR_t$ for a policy at time t in the following way:

$$IBNR_t = \mathbb{E}\left[P_{T_3} 1_{T_1 < t \wedge (T_0 + \Delta)} | \mathcal{A}_t\right]$$

where

$$\mathcal{A}_t = \{t < T_2, (F_u)_{T_0 \leq u \leq t}, (E_u)_{0 \leq u \leq t}\}.$$

- If $T_2 < t < T_3$, the claim has been reported to the company but the final assessment is still missing. Typically, we are in the situation where more and more information about the individual claim arrives, and the prediction uncertainty in the final assessment decreases. However, this claim is not completely settled, and therefore it is called a Reported But Not Settled (RBNS) claim. We can use the reporting date, the occurrence date, the risk factors, the claim history information as well as external information to predict the cumulated payment from t to the settlement date. We hence define the $RBNS_t$ for a policy at time t

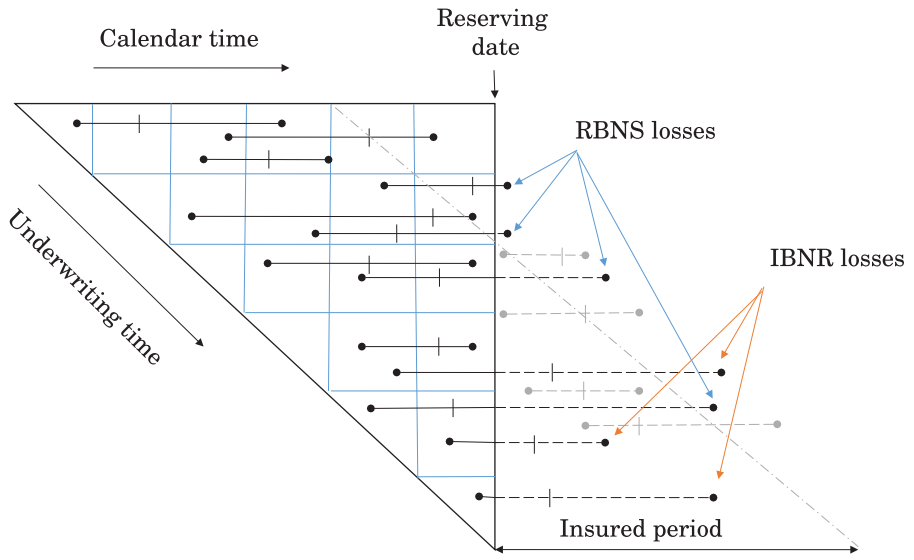

Fig. 3.2 Graphical representation of *incurred but not reported* (IBNR) and *reported but not settled* (RBNS) claims

for which a claim has been reported in the following way:

$$RBNS_t = \mathbb{E}\left[P_{T_3} - P_t | \mathcal{B}_t\right]$$

where

$$\mathcal{B}_t = \{T_2 < t < T_3, T_1 < T_0 + \Delta, (F_u)_{T_0 \leq u \leq t}, (E_u)_{0 \leq u \leq t}, (I_u)_{T_2 \leq u \leq t}, (P_u)_{T_1 \leq u \leq t}\}.$$

The individual claim reserve is eventually defined as

$$ICR_t = IBNR_t 1_{t<T_2} + RBNS_t 1_{t \geq T_2}.$$

Examples of IBNR and RBNS claims are depicted on Figure 2.

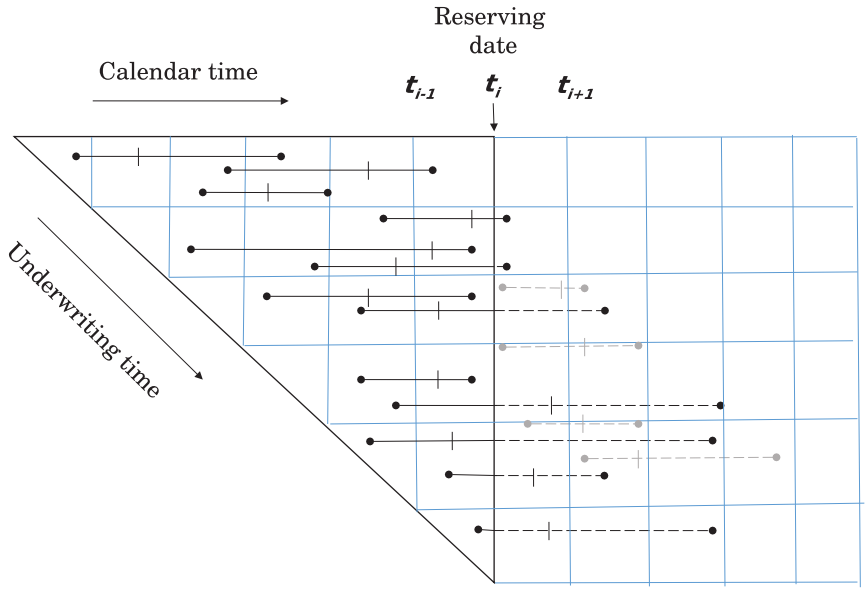

Fig. 3.3 The grid of times $(t_i)_{i\geq 0}$

3.2.2 Subdivisions of the outstanding claims over development periods

We now assume that there exists a maximum delay $\Delta_{\max,r}$ to report the claim once it has occurred, i.e. $T_2 - T_1 < \Delta_{\max,r}$, and that there exists a maximum delay $\Delta_{\max,s}$ to settle the claim once it has been declared, i.e. $T_3 - T_2 < \Delta_{\max,s}$. These assumptions can be easily weakened and are only used to write the outstanding claims reserves as finite sums.

As for the Chain Ladder approach that used several development periods, we will consider a grid of times $t_j = \delta \times j$, where $j \geq 0$ and δ is a fixed timestep. We will assume that the reserving date belongs to this grid and is given by t_i (see Figure 3). We then focus on subdivisions of the outstanding claims over the development periods $(t_{i+j-1}, t_{i+j}]$ for $j \geq 1$.

We define $RBNS_{t_i,j}$, $j = 1, 2, 3, ...$, as the expected increase of the payment process between t_{i+j-1} and t_{i+j} given that a claim has been declared and given all the information available on the policyholder and the claim history:

$$RBNS_{t_i,j} = \mathbb{E}\left[P_{t_{i+j}} - P_{t_{i+j-1}} | \mathcal{B}_{t_i}\right]$$

and it follows that

$$RBNS_{t_i} = \sum_{j=1}^{\lceil \Delta_{\max,s}/\delta \rceil} RBNS_{t_i,j}.$$

In the same way, we split $IBNR_{t_i}$ in the following way:

$$IBNR_{t_i,j} = \mathbb{E}\left[(P_{t_{i+j}} - P_{t_{i+j-1}})1_{T_1<t_i\wedge(T_0+\Delta)}|\mathcal{A}_{t_i}\right], \qquad j = 1, 2, 3, ...$$

such that

$$IBNR_{t_i} = \sum_{j=1}^{\lceil (\Delta_{\max,r}+\Delta_{\max,s})/\delta \rceil} IBNR_{t_i,j}.$$

Moreover we break down $IBNR_{t_i,j}$ trough a frequency/severity formula:

$$IBNR_{t_i,j} := IBNR\_freq_{t_i,j} \times IBNR\_loss_{t_i,j}$$

where

$$IBNR\_freq_{t_i,j} = \mathbb{E}\left[1_{(P_{t_{i+j}}-P_{t_{i+j-1}})1_{T_1<t_i\wedge(T_0+\Delta)}>0}|\mathcal{A}_{t_i}\right]$$

and

$$IBNR\_loss_{t_i,j} = \mathbb{E}[(P_{t_{i+j}}-P_{t_{i+j-1}})1_{T_1<t_i\wedge(T_0+\Delta)}|\mathcal{A}_{t_i}, (P_{t_{i+j}}-P_{t_{i+j-1}})1_{T_1<t_i\wedge(T_0+\Delta)} > 0].$$

The quantities $RBNS_{t_i,j}$, $IBNR\_loss_{t_i,j}$, resp. $IBNR\_freq_{t_i,j}$, for $j = 1, 2, 3, ...$, will be estimated using Machine Learning algorithms that will be trained with the quadratic loss function, resp. logistic loss, on specific train tests. We explain the building of these data sets in the next section.

3.3 Estimation via Machine Learning

3.3.1 Test sets construction

Let us first recall that the reserving date is denoted by t_i. The sets of policies for which $RBNS_{t_i}$ and $IBNR_{t_i}$ have to be evaluated are then given respectively by

$$\mathcal{P}_{te,RBNS} = \{p : T_{2,p} \leq t_i < T_{3,p}, T_{1,p} < T_{0,p} + \Delta\}$$

and

$$\mathcal{P}_{te,IBNR} = \{p : t_i < T_{2,p} \wedge (T_{0,p} + \Delta + \Delta_{\max,r} + \Delta_{\max,s})\}.$$

To estimate $RBNS_{t_i,j}$, $j = 1, 2, 3, ...$ for a policy p in $\mathcal{P}_{te,RBNS}$, we will use the following vector of variables

$$\left(T_{0,p}, t_i - T_{0,p}, F_{t_i,p}, E_{T_{0,p}}, E_{T_{1,p}}, E_{T_{2,p}}, I_{t_i,p}\right).$$

This vector contains the underwriting date, the exposure of the policy from the underwriting date to the reserving date, the risk factors associated to the policy valued at t_i, the external information observed at the underwriting date, the occurrence date and the reporting date, and the claim history up to t_i.

To estimate $IBNR\_freq_{t_i,j}$ and $IBNR\_loss_{t_i,j}$, $j = 1, 2, 3, ...$ for a policy p in $\mathcal{P}_{te,IBNR}$, we will only use the underwriting date, the exposure of the policy from the underwriting date to the reserving date, the risk factors associated to the policy valued at t_i, the external information observed at the underwriting date,

$$\left(T_{0,p}, t_i - T_{0,p}, F_{t_i,p}, E_{T_{0,p}}\right)$$

since the claim has not yet been reported to the insurance company (if ones occurs).

The matrices

$$\begin{aligned} \mathrm{X}_{te,RBNS} &= \left(T_{0,p}, t_i - T_{0,p}, F_{t_i,p}, E_{T_{0,p}}, E_{T_{1,p}}, E_{T_{2,p}}, I_{t_i,p}\right)_{p\in\mathcal{P}_{te,RBNS}} \\ \mathrm{X}_{te,IBNR} &= \left(T_{0,p}, t_i - T_{0,p}, F_{t_i,p}, E_{T_{0,p}}\right)_{p\in\mathcal{P}_{te,IBNR}} \end{aligned}$$

are referred to as the X.TEST sets.

In Figure 4, X.TEST is built by using covariates of policies whose histories begin in the red triangle. Figure 4 illustrates the case of predictions for $j = 1$ and Y.TEST (represented as a green rectangle) then contains the increases of payments (or the indicator functions of a positive increase of payments) between t_i and t_{i+1} for all policies in $\mathcal{P}_{te,RBNS}$ or in $\mathcal{P}_{te,IBNR}$.

It is noteworthy that there is no prediction to make for the covariates with this procedure. At time t_i and for each development period j, the X.TEST for the RBNS reserves uses $T_{0,p}$, $t_i - T_{0,p}$, $F_{t_i,p}$, $E_{T_{0,p}}$, $E_{T_{1,p}}$, $E_{T_{2,p}}$, $I_{t_i,p}$, and, for the RBNS reserves, $T_{0,p}$, $t_i - T_{0,p}$, $F_{t_i,p}$, $E_{T_{0,p}}$. All these covariates are observed at time t_i. We shall explain in the following subsection how to put these covariates that are observed at time $t_{i-j-k+1}$, with $j \geq 1$ and $k \geq 1$ in the train sets such that they provide the same type of information as for the test sets.

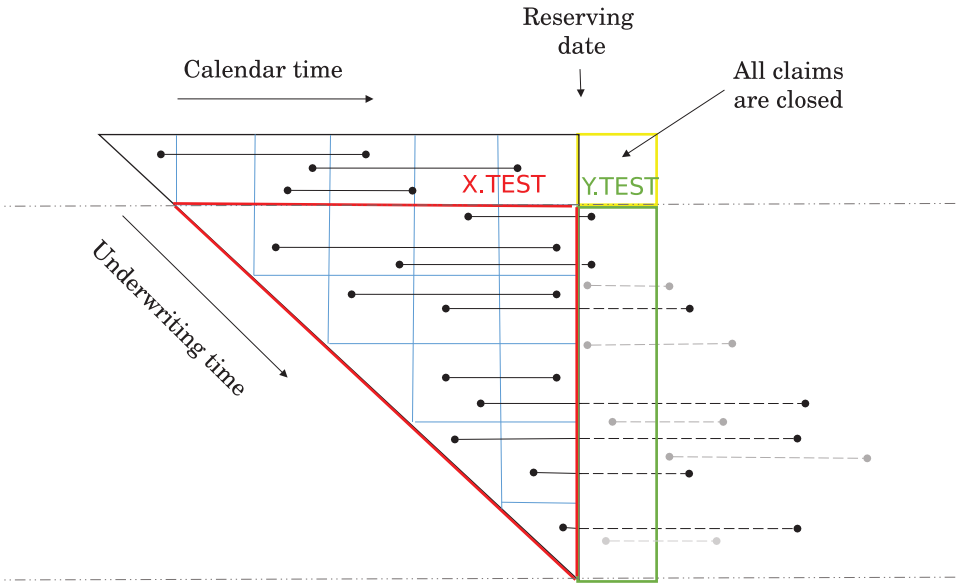

Fig. 3.4 Graphical representation of X.TEST and Y.TEST for $j = 1$

3.3.2 Train sets construction

For each development period j, we consider several prediction models that will be trained on different train sets. We therefore begin by defining three subsets of policies for each development period j and each model k:

- for $RBNS_{t_i,j}$

$$\mathcal{P}^{(j,k)}_{tr,RBNS} = \{p : T_{2,p} \leq t_{i-j-k+1} < T_{3,p}, T_{1,p} < T_{0,p} + \Delta\},$$

- for $IBNR\_freq_{t_i,j}$

$$\mathcal{P}^{(j,k)}_{tr,IBNR\_freq} = \{p : t_{i-j-k+1} < T_{2,p} \wedge (T_{0,p} + \Delta + \Delta_{\max,r} + \Delta_{\max,s})\},$$

- for $IBNR\_loss_{t_i,j}$

$$\mathcal{P}^{(j,k)}_{tr,IBNR\_loss} = \{p : t_{i-j-k+1} < T_{2,p}, (P_{t_{i-k+1}} - P_{t_{i-k}})1_{T_{1,p} < t_{i-j-k+1} \wedge (T_{0,p}+\Delta)} > 0\}.$$

We associate to $\mathcal{P}^{(j,k)}_{tr,RBNS}$, a X.TRAIN set defined by

$$\mathrm{X}^{(j,k)}_{tr,RBNS} = \Big(T_{0,p}, t_{i-j-k+1} - T_{0,p}, F_{t_{i-j-k+1},p}, E_{T_{0,p}}, E_{T_{1,p}}, E_{T_{2,p}}, I_{t_{i-j-k+1},p}\Big)_{p\in\mathcal{P}^{(j,k)}_{tr,RBNS}}$$

and a Y.TRAIN set defined by

$$\mathrm{Y}^{(j,k)}_{tr,RBNS} = \Big(P_{t_{i-k+1},p} - P_{t_{i-k},p}\Big)_{p\in\mathcal{P}^{(j,k)}_{tr,RBNS}}.$$

We associate to $\mathcal{P}^{(j,k)}_{tr,IBNR\_freq}$, a X.TRAIN set defined by

$$\mathrm{X}^{(j,k)}_{tr,IBNR\_freq} = \Big(T_{0,p}, t_{i-j-k+1} - T_{0,p}, F_{t_{i-j-k+1},p}, E_{T_{0,p}}\Big)_{p\in\mathcal{P}^{(j,k)}_{tr,IBNR\_freq}}$$

and a Y.TRAIN set defined by

$$\mathrm{Y}^{(j,k)}_{tr,IBNR\_freq} = \Big(1_{(P_{t_{i-k+1,p}}-P_{t_{i-k,p}})1_{T_{1,p}<t_{i-j-k+1}\wedge(T_{0,p}+\Delta)>0}}\Big)_{p\in\mathcal{P}^{(j,k)}_{tr,IBNR\_freq}}.$$

We associate to $\mathcal{P}^{(j,k)}_{tr,IBNR\_loss}$, a X.TRAIN set defined by

$$\mathrm{X}^{(j,k)}_{tr,IBNR\_loss} = \Big(T_{0,p}, t_{i-j-k+1} - T_{0,p}, F_{t_{i-j-k+1},p}, E_{T_{0,p}}\Big)_{p\in\mathcal{P}^{(j,k)}_{tr,IBNR\_loss}}$$

and a Y.TRAIN set defined by

$$\mathrm{Y}^{(j,k)}_{tr,IBNR\_loss} = \Big(P_{t_{i-k+1,p}} - P_{t_{i-k,p}}\Big)_{p\in\mathcal{P}^{(j,k)}_{tr,IBNR\_loss}}.$$

Figure 5 (resp. 6, 7) illustrates the case $j = 1$, $k = 1$ (resp. $j = 1$, $k = 2$ and $j = 2$, $k = 1$).

3.3.3 Final claim reserves prediction

We will denote the vectors of predictions for development period j and prediction model k as

$$\begin{aligned}
\hat{\mathrm{Y}}^{(j,k)}_{te,RBNS} &= (\widehat{RBNS}^{(k)}_{t_i,j,p})_{p\in\mathcal{P}_{te,RBNS}} \\
\hat{\mathrm{Y}}^{(j,k)}_{te,IBNR_freq} &= (\widehat{IBNR_freq}^{(k)}_{t_i,j,p})_{p\in\mathcal{P}_{te,IBNR}} \\
\hat{\mathrm{Y}}^{(j,k)}_{te,IBNR_loss} &= (\widehat{IBNR_loss}^{(k)}_{t_i,j,p})_{p\in\mathcal{P}_{te,IBNR}}
\end{aligned}$$

where k denotes the prediction model associated to the k-th train sets.

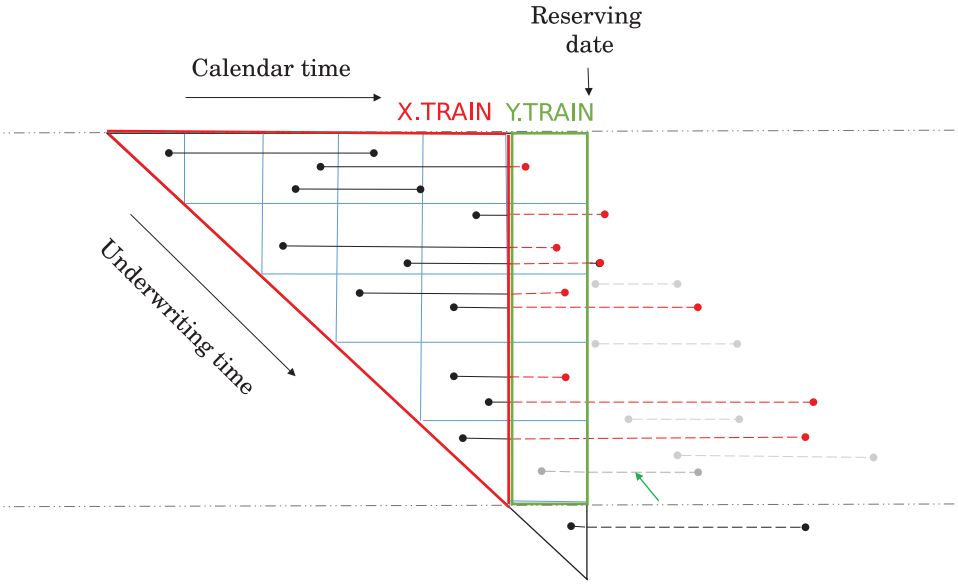

Fig. 3.5 Graphical representation of X.TRAIN and Y. TRAIN for $j = 1$, $k = 1$

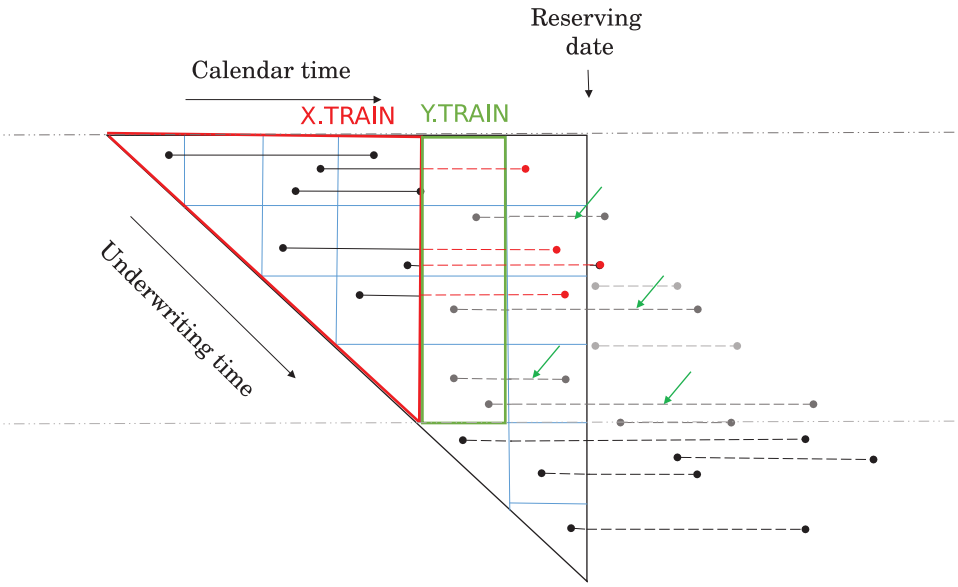

Fig. 3.6 Graphical representation of X.TRAIN and Y. TRAIN for $j = 1$, $k = 2$

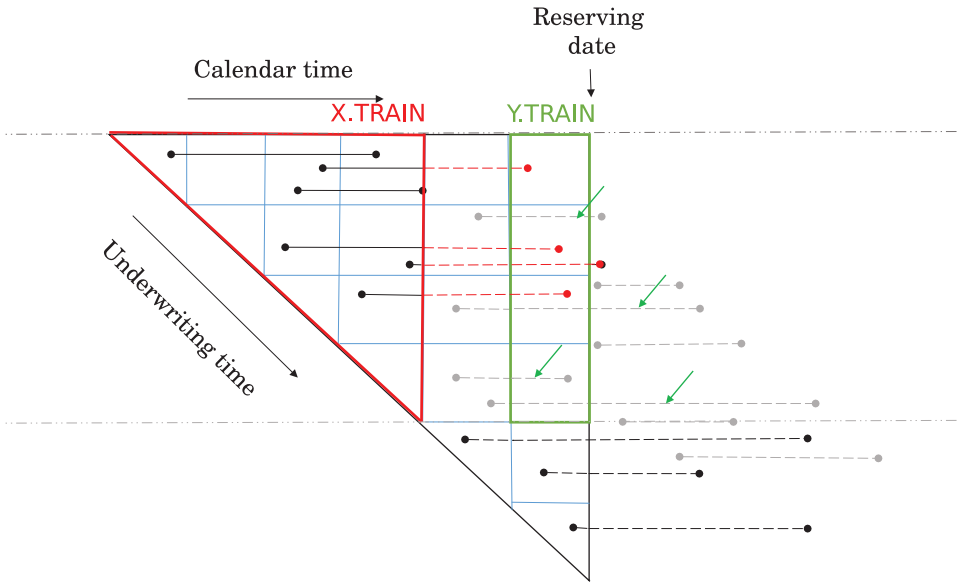

Fig. 3.7 Graphical representation of X.TRAIN and Y. TRAIN for $j = 2$, $k = 1$

For a policy p in $\mathcal{P}_{te,RBNS}$, we average predictions over the different models considered:

$$\widehat{RBNS}_{t_i,j,p} = \sum_k p_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right) \widehat{RBNS}^{(k)}_{t_i,j,p}$$

where $\left(p_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right)\right)_k$ are sets of positive weights such that

$$\sum_k p_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right) = 1 \text{ for each j=1,2,...}$$

In the same way, for p in $\mathcal{P}_{te,IBNR}$, we average predictions over the different models considered:

$$\begin{aligned}
\widehat{IBNR_freq}_{t_i,j,p} &= \sum_k q_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right) \widehat{IBNR_freq}^{(k)}_{t_i,j,p} \\
\widehat{IBNR_loss}_{t_i,j,p} &= \sum_k r_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right) \widehat{IBNR_loss}^{(k)}_{t_i,j,p}
\end{aligned}$$

where $\left(q_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right)\right)_k$ and $\left(r_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right)\right)_k$ are sets of positive weights such that $\sum_k q_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right) = 1$ and $\sum_k r_{j,k}\left(t_{i-k+1}, E_{t_{i-k+1}}\right) = 1$ for each $j = 1, 2, \ldots$. It is advised to use decreasing weights with respect to k to get more responsive

aggregated models in case of non-stationarity of the generating process. If one wants to take into account information based on E, the exercise is actually more complex since weights should themselves be determined by an additional learning (train/test) task that uses a more complex cross-validation approach.

We denote by $ICR_{t_i,p}$ the claim reserve, at date t_i, for a policy p. The estimation of this quantity is computed by summing IBNR and RBNS reserves for p:

$$\widehat{ICR}_{t_i,p} = \widehat{IBNR}_{t_i,p} 1_{t_i < T_{2,p}} + \widehat{RBNS}_{t_i,p} 1_{t_i \geq T_{2,p}}$$

where

$$\widehat{IBNR}_{t_i,p} = \sum_{j=1}^{\lceil (\Delta_{\max,r} + \Delta_{\max,s})/\delta \rceil} \widehat{IBNR\_freq}_{t_i,j,p} \widehat{IBNR\_loss}_{t_i,j,p}$$

and

$$\widehat{RBNS}_{t_i,p} = \sum_{j=1}^{\lceil \Delta_{\max,s}/\delta \rceil} \widehat{RBNS}_{t_i,j,p}.$$

We can finally compute the global claims reserve of the portfolio by summing all the reserves predictions:

$$\sum_{p \in \mathcal{P}_{te,RBNS} \cup \mathcal{P}_{te,IBNR}} \widehat{ICR}_{t_i,p}$$

3.4 Estimation via Chain Ladder

In this section we present the famous Chain Ladder technique used by actuaries, such that the reader is able to compare it with our approach. The first step is to aggregate claims data in a development triangle organized by origin period (occurrence time) and development period. Since we have considered until now the underwriting time as the second axis in our graphical representation (see Figure 1), we must project the claim history lines on this axis such that it now depicts the occurrence time (see Figure 8).

The second step of the Chain Ladder technique is to compute the development factors for $j = 1, .., J$ where $J = \lceil (\Delta + \Delta_{\max,r} + \Delta_{\max,s})/\delta \rceil$. The set of policies to compute the j-th development factor at time t_i is characterized by:

$$\mathcal{P}_{tr,CL}^{(j|i)} = \{p : T_{2,p} \leq t_{i-j}\}.$$

For a policy p, let $T_{1,p}^{(\delta)} = \inf_{t_j \geq T_{1,p}} t_j$ be the smallest t_j greater than $T_{1,p}$, i.e. the smallest time on the time grid greater than the occurrence date of the claim of the

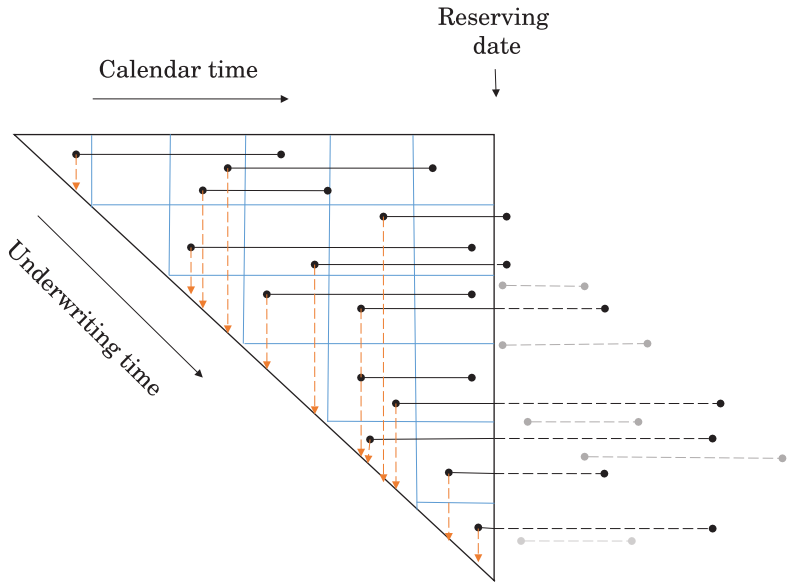

Fig. 3.8 From underwriting time to occurrence time

policy. Then the j-th development factor is computed as

$$\hat{F}_{j|i} = \frac{\sum_{p \in \mathcal{P}_{tr,CL}^{(j|i)}} P_{T_{1,p}^{(\delta)}+\delta j,p}}{\sum_{p \in \mathcal{P}_{tr,CL}^{(j|i)}} P_{T_{1,p}^{(\delta)}+\delta(j-1),p}}.$$

Figure 9 (resp. 10) illustrates the computation for the first (resp second) development factor. The numerator of $\hat{F}_{1|i}$ (resp. $\hat{F}_{2|i}$) is the sum of the payments from the occurrence time to $T_{1,p}^{(\delta)} + \delta$ (resp. $T_{1,p}^{(\delta)} + 2\delta$) over $\mathcal{P}_{tr,CL}^{(1|i)}$ (resp. $\mathcal{P}_{tr,CL}^{(2|i)}$) (green squares). The denominator is the sum of the payments from the occurrence time to $T_{1,p}^{(\delta)}$ (resp. $T_{1,p}^{(\delta)} + \delta$) over $\mathcal{P}_{tr,CL}^{(1|i)}$ (resp $\mathcal{P}_{tr,CL}^{(2|i)}$) (red squares).

The second step of the Chain Ladder technique is to compute the predictions of the sum the payments step by step by using recursively the previously estimated development factors. The set of policies for which the predictions used the j-th development factor at time t_i is given by:

$$\mathcal{P}_{te,CL}^{(i,j)} = \{p : t_{i-j} \leq T_{2,p} \leq t_i\}\}.$$

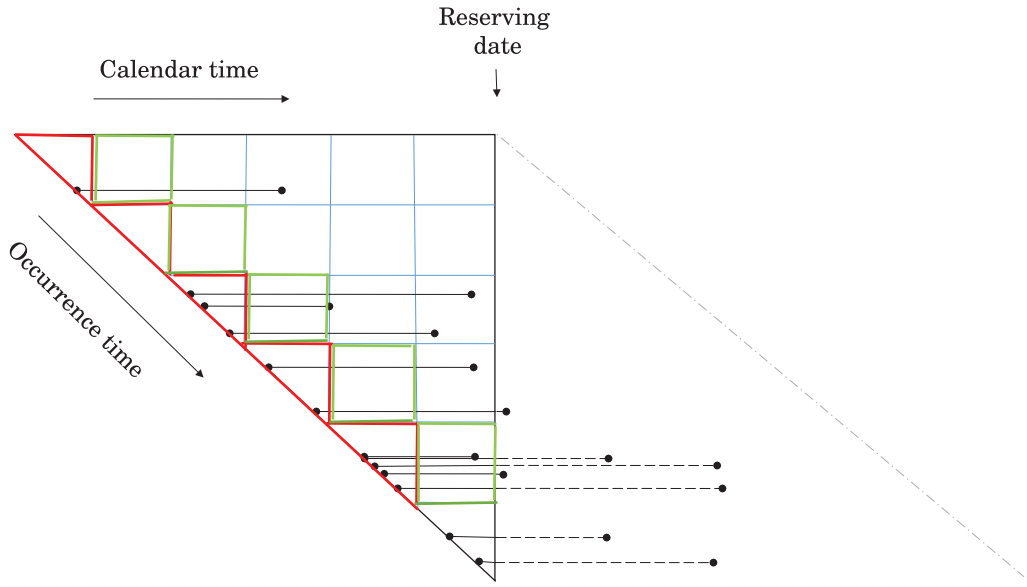

Fig. 3.9 Graphical representation of the first development period for CL method

Figure 10 (resp. 12) illustrates the computation for the predictions that use the first (resp second) development factor.

The final claims reserve prediction for the whole portfolio with the Chain Ladder method is therefore given by:

$$\sum_{j=1}^{J}\left(\sum_{p\in\mathcal{P}_{te,CL}^{(i,j)}} P_{T_{1,p}^{(\delta)}+\delta(j-1),p}\right)\times\left(\prod_{k=1}^{J-j}\hat{F}_{k+(j-1)|i}-1\right).$$

Let us discuss the main underlying technical assumption of such a prediction. The j-th individual development factor for the accident period $(t_{k-1},t_k]$ is defined as

$$\hat{f}_{j|k}=\frac{\sum_{p\in\mathcal{R}_{tr,CL}^{(k)}} P_{T_{1,p}^{(\delta)}+\delta j,p}}{\sum_{p\in\mathcal{R}_{tr,CL}^{(k)}} P_{T_{1,p}^{(\delta)}+\delta(j-1),p}}$$

where $\mathcal{R}_{tr,CL}^{(k)}=\{p: t_{k-1}\leq T_{1,p}\leq t_k\}$. The Chain Ladder predictions are based on the assumption that the individual development factors do not depend on the accident

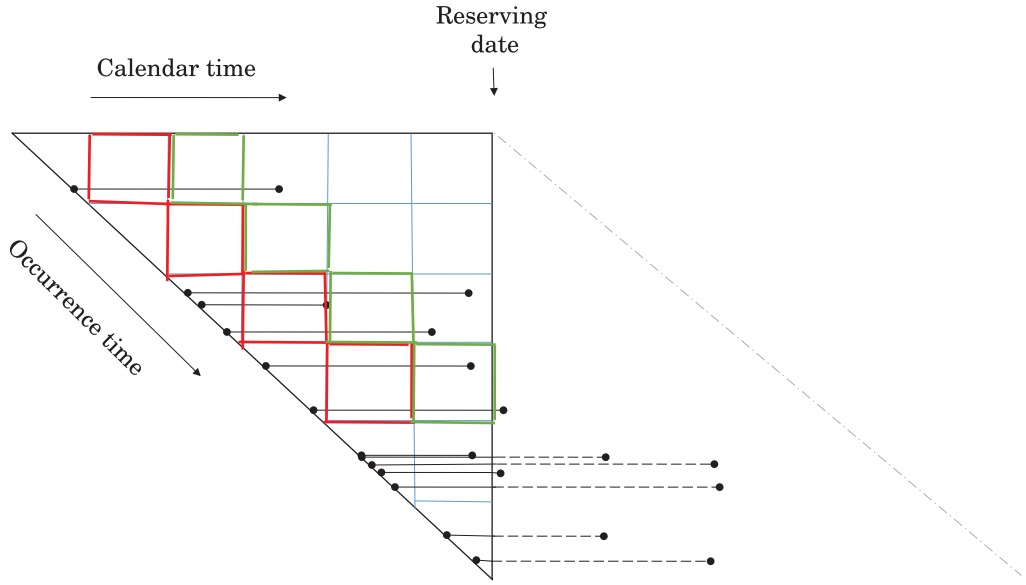

Fig. 3.10 Graphical representation of the second development period for CL method

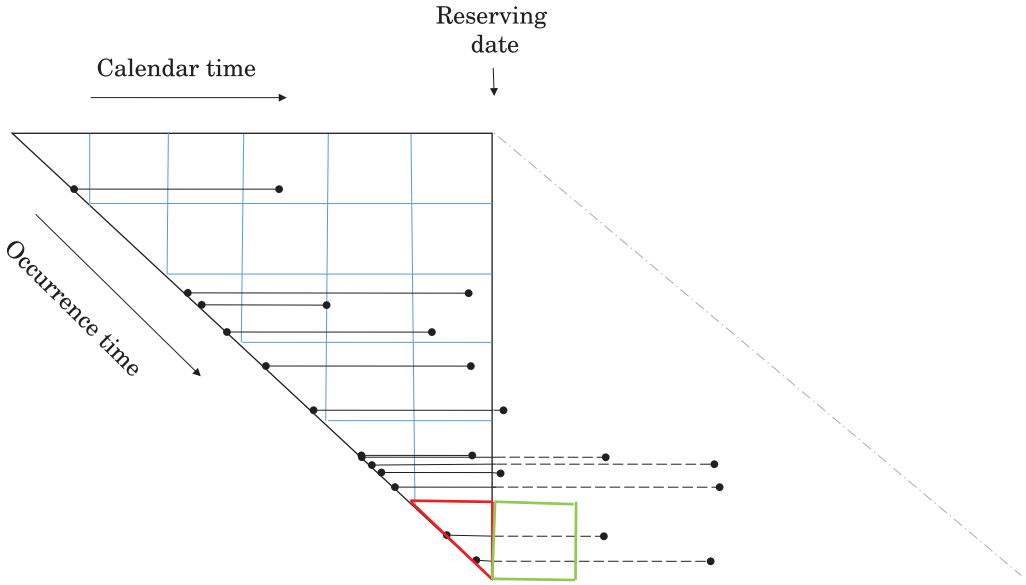

Fig. 3.11 Graphical representation for the predictions of the first development period

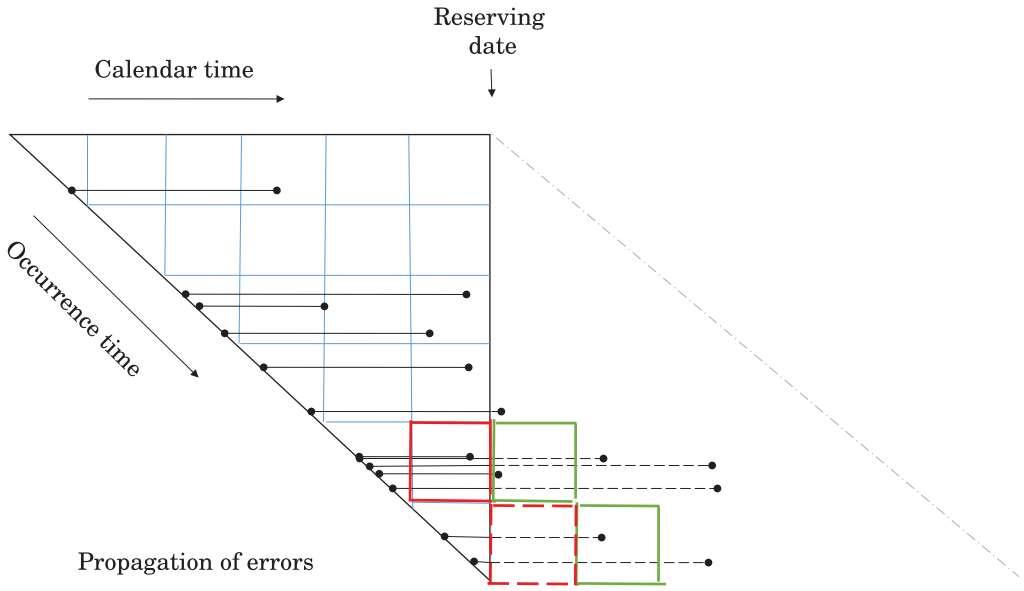

Fig. 3.12 Graphical representation for the predictions of the second development period

periods and are roughly equal to a constant f_j, or equivalently that for $k = 1, 2, ..., i$, $\hat{f}_{j|k} \approx f_j$. Therefore the Chain Ladder method is appropriate when loss developments patterns have been historically stable and when they can be reasonably assumed as not to be varying in the future. There are plenty of reasons for which this assumption would not hold. For example, if the underwriting date (or the occurrence date) has an impact on the payment delay or on the reporting delay, this assumption is clearly broken. When comparing alternative prediction methods to the Chain Ladder, it is important to keep this point in mind. It is well-known by actuaries that they should be very cautious from relying solely on standard Chain Ladder techniques when there could be changes in the developments patterns of claims due e.g. to significant company operational changes or also to important changes in the insurance regulatory, legislative, or judicial landscapes.

It is also noteworthy to say that the choice of the aggregation level in the Chain Ladder method could be very important and that it is in general discussed neither from a theoretical point of view nor from a practical point of view. In practice aggregation is usually done in years because the reliability of the data is higher at the end of the calendar year for accountability reasons. Here we have decided to take the same

aggregation level, δ, as the one used for characterizing the fixed timesteps of the grid of times for the Machine Learning approach in order to get comparable results.

3.5 Simulation study - mobile phone insurance

In this section, we illustrate our method on a synthetic insurance portfolio. Although a detailed real data study would be better, it is very hard (even impossible) to get enough individual data from a single insurance portfolio to have accurate assessments of the bias and variances of the estimators. In some sense a synthetic insurance portfolio appears as a compromise between reality and ability to get lots of data.

Most people now have a mobile phone and it has become an essential part of everyday life. Losing or damaging their mobile phone can be a total nightmare and they may turn down to insurance coverage. For this case study, we consider a mobile phone insurance that covers the devices in the event of theft, breakage or oxidation. The insurance company provides cover for an insured period of one year and for a range of four brands and up to four models by brand with three policy types available: "breakage", "breakage and oxidation" and "breakage, oxidation and theft".

Such an insurance product has short-tailed lines compared to e.g. a general liability contract. It does not present a large risk for the insurance company as soon as it can be assumed that it has been sold directly to the customer in a perfect environment where there would be any asymmetric information and no moral hazard and adverse selection risk. In practise, this is not the case since such a contract is linked to a service or product distributed by a company (not always in insurance business), which is not the main customer's purchase motive (it is called an affinity insurance product). Such a product could be very risky due to adverse behaviors from policyholders and should be monitored with care by the insurance company.

To illustrate our method and compare it to the Chain Ladder approach, we will first consider a central scenario with very simplified portfolio assumptions (in particular they guarantee that the individual development factors of the Chain Ladder method are approximately constant over the accident periods). We will then study several scenarii around this central scenario for which we will modify one of these simplified assumptions (and the assumptions of the Chain Ladder method will no more necessarily hold). Such alternative scenarii for which there is one modification of the "ideal" assumptions are also used to take into account potential adverse behaviors from policyholders.

The underwriting period will be from the first day of 2016 to the last day of 2017 (after this date, the portfolio starts its run-off period). We will perform the reserving

exercises for each end of month from September 2016 to July 2017 and will compute IBNR and RBNS claims reserves with our Machine Learning method, and the global claims reserves with the Chain Ladder method. We will compare these computed reserves with the true reserves and study forecast bias and variance for both methods. Note that the timestep we fixed for this illustration is a uniform monthly timestep (actually we assume that each month has 30 days and that a year has then 360 days). We discuss what happens when a true calendar is used in an additional scenario.

The aim of this section is to discuss the "Best Estimate" loss reserving and not the "Risk Margin" computation (as defined by the Solvency II directive). That is why we compare the mean values of the predictions to evaluate bias of the predictions of the Best Estimate values and the standard errors (or equivalently 95% confidence intervals under the assumption of Gaussian distributions) to have an idea of their accurateness, but not to discuss the reserve uncertainty. An additional algorithm is actually needed to estimate conditional quantiles of the predictive distributions and an appropriate bootstrap strategy has to be proposed.

3.5.1 Technical assumptions of the central scenario

The central scenario aims at being very simple in order to benchmark the Chain Ladder and Machine Learning methods when the portfolio is in an almost stationary environment. We now present its assumptions.

The underwriting Poisson point process has a constant intensity $\lambda_0(t) = 700$ (daily basis), i.e. the insurance company sells roughly 500,000 insurance policies over the two underwriting years. Each policy is generated independently from the other policies.

The three available policy types are assumed to be sold according to the following constant over time distribution

	Probability
Breakage	0.25
Breakage + oxidation	0.45
Breakage + oxidation + theft	0.30

Table 1: Policy types distribution

The policyholders are assumed to insure mobile phone whose brand is distributed according to the following constant over time distribution (note that the basis price of a mobile phone also depends on its brand)

	Probability	Basis price
Brand 1	0.45	600
Brand 2	0.30	550
Brand 3	0.15	300
Brand 4	0.10	150

Table 2: Brand distribution

The price of a mobile phone is equal to a basis price multiplied by a factor that depends on the model type according to the following table (Table 3 also provides the model type distribution)

Model type	Multiplicative factor	Probability
0	1	0.05
1	1.15	0.10
2	1.15^2	0.35
3	1.15^3	0.50

Table 3: Model type distribution and multiplicative factors with respect to the model types

The policy type, the mobile phone brand and the mobile phone model type are independent from each other.

We assume that claim frequencies are generated through a competing model between risks, with constant incidence hazard rates that can depend on the mobile phone model type (but neither on the brand nor the model type), in the following way

	Yearly incidence rate
Breakage	0.15
Oxidation	0.05
Theft	0.05× model type

Table 4: Incidence hazard rates

We assume that the claim amount is a percentage of the buying price of the mobile phone and the distribution of this percentage is a Beta distribution with parameters depending on the claim type

	α	β
Breakage	2	5
Oxidation	5	3
Theft	5	0.5

Table 5: Parameters of the Beta distributions

Once a claim occurs, we assume that the reporting delay hazard rate is given by

$$\lambda_1(t+T_0) = \frac{t^{\alpha-1}\,(1-t)^{\beta-1}}{\int_t^1 u^{\alpha-1}\,(1-u)^{\beta-1}\,du}, \qquad 0<t<1,$$

with $\alpha = 0.4$, $\beta = 10$ (the mean reporting delay is roughly 14 days). Claims are settled in one payment and the payment delay hazard rate is given by

$$\lambda(t+T_1) = \frac{((t-d)/m)^{\alpha-1}\,(1-(t-d)/m)^{\beta-1}}{m\int_{(t-d)/m}^1 u^{\alpha-1}\,(1-u)^{\beta-1}\,du}, \qquad d<t<m+d,$$

with $\alpha = 7$, $\beta = 7$, and with $m = 40/360$, $d = 10/360$ (the mean payment delay is roughly 30 days). These hazard rates do not depend neither on the brand, the model, the coverage type, nor the occurrence date.

3.5.2 Selected features

The main strength of our Machine Learning approach is to take into account any information about the policy (for IBNR and RBNS claims), and the claim history (for RBNS claims). No external information is considered here. For this case study, we use the following policy-related features (embedded in F_t):

- phone brand,
- phone price,
- phone model type,
- coverage type ("Breakage", "Breakage and Oxidation", "Breakage and Oxidation and Theft").

We moreover use the following claim-related features (embedded in I_t):

- type of damage ("Breakage", "Oxidation", "Theft"),
- reporting delay (in days)
- number of days since the claim has been declared.

In case of transitional payments, additional features could be the amount paid and the number of payments made until the reserving date.

3.5.3 Algorithm used for prediction

Once the train sets have been built (see Section 3), we can train a prediction algorithm. For this case study, we decided to illustrate our methodology by using the ExtraTrees algorithm (Geurts et al. (2006)). This algorithm builds an ensemble of unpruned regression trees according to a classical top-down procedure. Its two main differences with other tree-based ensemble methods are:

1. it splits nodes by choosing cut-points fully at random;
2. it uses the whole learning sample (rather than a bootstrap replica) to grow the trees.

The predictions of the trees are aggregated to yield the final prediction, by majority vote in classification problems and arithmetic average in regression problems. From the bias-variance point of view, the rationale behind the Extra-Trees method is that the explicit randomization of the cutpoint and attribute combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes used by other methods. The usage of the full original learning sample rather than bootstrap replicas is motivated in order to minimize bias. From the computational point of view, the complexity of the tree growing procedure is like most other tree growing procedures.

Note however that the algorithm used for prediction is not the keystone of our method since we can use other Machine Learning algorithm to make predictions such as RandomForests or Gradient Boosting algorithms like XGBoost (which can make slightly better predictions than ExtraTrees).

3.5.4 Results in the central scenario

We performed 40 simulations of the portfolio over the two years. For each end of month from September 2016 to May 2018, we computed the outstanding claims reserves with both predictive methods (40 times). Note that for the ML method we only used one model ($k = 1$).

We first averaged the true values of the sum of the cumulated payments called here Ground Truth (GT), the Machine Learning predictions (ML), and the Chain Ladder predictions (CL). These values are represented by solid lines on the left part of Figure 13.

We can observe that both methods are almost unbiased except from January 2017 to March 2017 for the ML method and to July 2017 for the CL method. January 2017

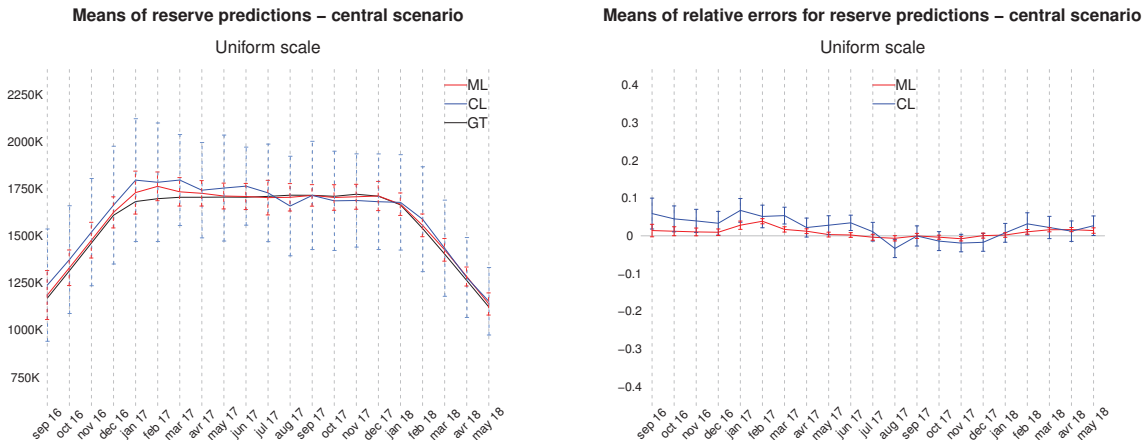

Fig. 3.13 Means of reserve predictions and mean relative errors for the machine learning (ML) and chain ladder (CL) methods — central scenario (the vertical dashed segments provide the 95% prediction spans for each prediction). GT, ground truth

is actually the first month where the portfolio size and its exposure to risk has begun to be stabilized (i.e. underwritings and terminations are balanced). Both algorithm need time to adjust their predictions, but the ML method does it more quickly.

The vertical dashed segments provide the 95% prediction spans for each prediction (they are based on the assumption of a Gaussian distribution, i.e. their lengths are approximately equal to four times the standard errors computed over the 40 simulations). We can see that the standard errors of the ML predictions are really lower than the CL predictions (around 3 or 4 times smaller). So one first conclusion here is that the ML method outperforms the CL method in the variance-of-reserves point of view because it takes into account the heterogeneity between claims to provide more accurate predictions.

It should be underlined that the variances of the CL predictions are very close to the variances computed with the Mack's formulas (Mack (1999)) in this case study. Moreover we observed that the standard errors of the ML predictions are a little larger than the standard errors of the GT values (around 1.6 time larger). Note that the spans of the GT values have not been plotted.

On the right part of Figure 13, we plot the average values of the relative errors between the true values of the sum of the increases of payments and respectively the ML predictions and the CL predictions (ratios of the differences between both predictions and the GT over the GT). The vertical solid segments provide the 95% confidence intervals of the means of the relative errors and show that the bias of the

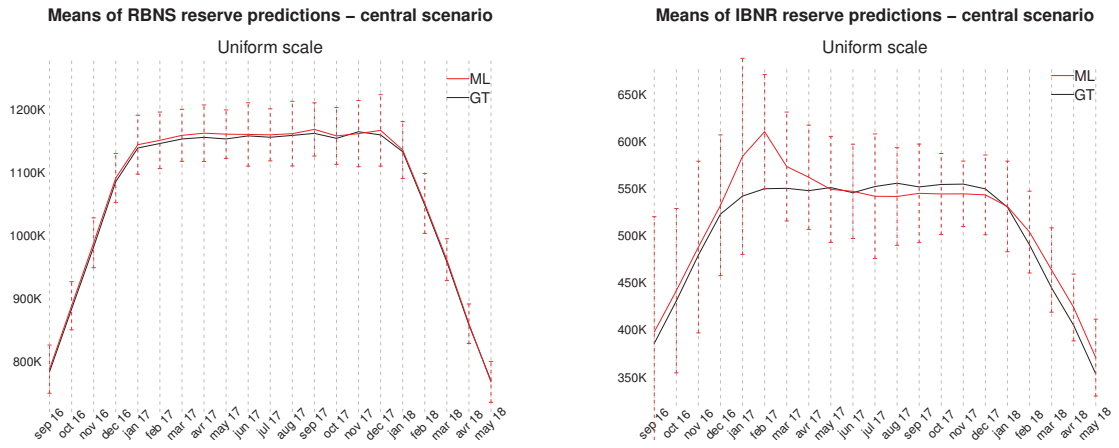

Fig. 3.14 Means of the *incurred but not reported* (IBNR) and *reported but not settled* (RBNS) reserve predictions for the machine learning (ML) method (the vertical dashed segments provide the 95% prediction spans for each prediction). GT, ground truth

relative errors are statistically significant for almost all CL predictions, but not for the ML predictions.

On the other hand, we plot on Figure 14 the averages of the ML RBNS reserves and of the ML IBNR reserves, as well as the averages of their respective true values. One can see that the ML RBNS reserves are really precise since they are unbiased and their standard errors are not very large. The ML IBNR reserves are however a bit biased when the portfolio size starts to be stabilized. This is due to the fact that when the algorithm learns exposures as of the first day of 2017, it has never seen that exposures are actually bounded since the insured period is equal to one year (this means that policies underwritten at the beginning of January 2016 are no more exposed to the whole month of January 2017, and the algorithm has never learnt such a behavior). This leads to a punctual mis-prediction.

3.5.5 The other scenarii

We now study several scenarii around this central scenario for which we will modify one (and only one) of the assumptions. The assumptions that we changed are the following ones:

- monthly scale instead of uniform scale,

- time-dependent underwriting rate,

- increase/decrease of 10% of the payment delay,
- arrivals of new phone models (more expensive) at the beginning of the year 2017,
- increase of 40% of the claim rate from mid-December 2016 to mid-January 2017,
- multiple payments.

Note that we also studied the scenario where the intensity of the underwriting Poisson point process is divided by 10 ($\lambda_0(t) = 70$). We did not write any sub-section of this scenario since the comparisons between both prediction methods gave qualitatively the same results. The lengths of 95% prediction spans are naturally larger, but let us underline all the same that the standard errors of the ML predictions are even lower than the CL predictions (around 4.5 times smaller).

3.5.5.1 Monthly scale

We first wanted to study the consequences of using a true calendar rather than using a uniform timestep and dividing a whole year in twelve equal parts (the timestep size on the central scenario was 30 days). On Figure 15, we plot the average values of the relative errors between the true values of the sum of the increases of payments and respectively the ML predictions and the CL predictions. We notice a strong month-dependent bias for the CL method (the assumption about a constant aggregation level compulsory for the CL method is not satisfied), but a moderate month-dependent bias for the ML method. These bias are the consequences of the fact that months do not have the same length, leading to a mis-learning of the true exposure as well as a mis-prediction for the true future exposure (although this last issue could be fixed by taking into account the true number of days of the month for which the predictions are made). These are interesting result from the insurer's point of view because the ML method can adapt quite well when the timestep length is not constant over time.

The vertical solid segments provide the 95% confidence intervals of the means of the relative errors and show that the bias are statistically significant for almost all CL predictions. We also see that the ML method has very shorter confidence intervals.

3.5.5.2 Time-dependent underwriting rate

In this scenario, we let the underwriting rate change over time: we multiplied the constant hazard rate by the density of a Beta distribution with $\alpha = 1.6$ and $\beta = 2.5$, scaled over the two years. Figure 16 shows the reserves evolution for the Ground Truth

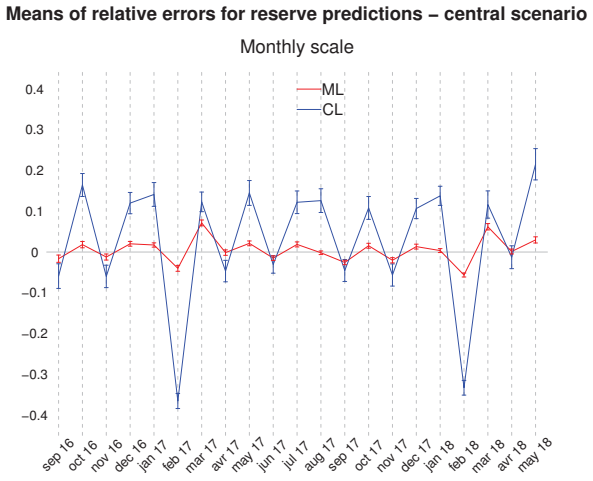

Fig. 3.15 Mean relative errors of reserve predictions for the machine learning (ML) and chain ladder (CL) methods (the vertical dashed segments provide the 95% prediction spans for each prediction)

(GT), the ML method and the CL method. We can see that both methods perform well, and we still observe a better standard error for the ML method as we observed for the central scenario.

Note that, although the underwriting rate depends on time, the distributions of the reporting delay and the payment delay are not modified and therefore the assumptions of the CL model are still satisfied.

3.5.5.3 Payment delay modification

In this scenario, we reduce (resp. increase) all the payments delays by 10% starting from the first day of 2017 until the total run-off of the portfolio. This means that a claim that was settled in 10 days will now be settled in 9 days (resp. 11 days). Hence the assumption of a stationary payment delay is no more satisfied and it is expected that the CL method fails. We can actually see from Figure 17 and 18 that this change of assumptions leads to a huge effect on the CL reserves. It is however not the case for the ML reserves. This is due to a combination of some reasons explained here:

- An increase of the number payments will be observed for the month of January 2017. This increase will immediately affects the computation of the first development factor of the CL method. And this factor will have a strong impact on

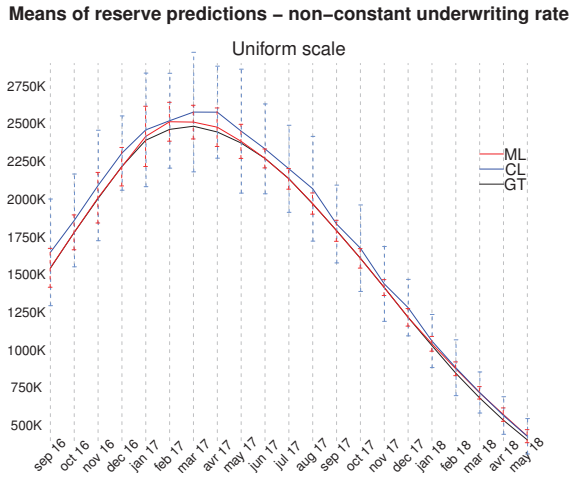

Fig. 3.16 Means of reserve predictions for the ML and CL methods - time-dependent underwriting rate (the vertical dashed segments provide the 95% prediction spans for each prediction)

the estimated values of the ultimate cumulated payments for the claims that have occurred and have been reported during the last month. This leads to an important over-estimation of the reserves if the computation of the CL method is done mechanically (which is fortunately not the case in practice, because such a result would alarm the actuary in charge of the reserving exercise). Since the ML prediction is not based on a multiplicative form (that leads to the propagation of the errors), this shock on the payment delay will be softer when using the ML method.

- This phenomenon is amplified by the fact that the average payment delay is around 30 days on the central scenario. This leads to a huge amount of claims which are paid in January compared to December.

In practice, it is the strong difference between the ML and CL computations that should be highlighted. It seems to provide an indicator of important changes whose source could be looked for in the payment delays.

On the other hand, we observe the opposite behavior if the payment delay is lengthened by 10% leading to a huge under-estimation of the reserves.

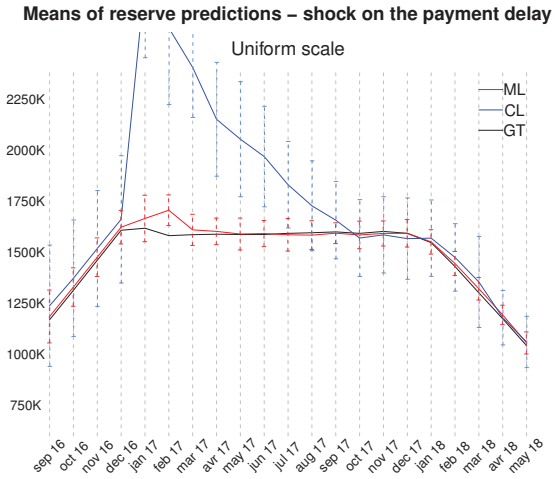

Fig. 3.17 Means of reserve predictions for the ML and CL methods - negative shock on the payment delay (the vertical dashed segments provide the 95% prediction spans for each prediction)

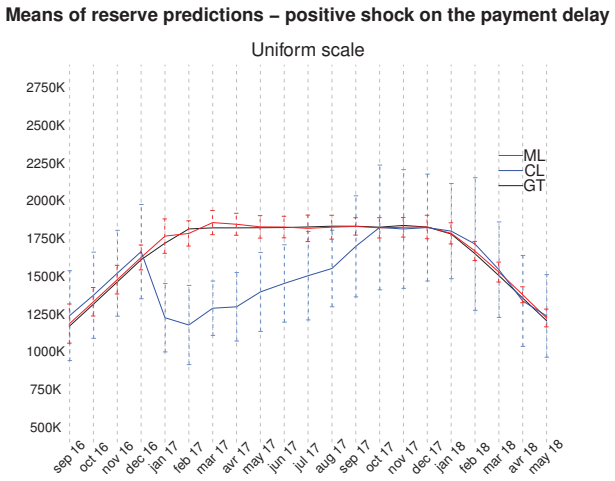

Fig. 3.18 Means of reserve predictions for the ML and CL methods - positive shock on the payment delay (the vertical dashed segments provide the 95% prediction spans for each prediction)

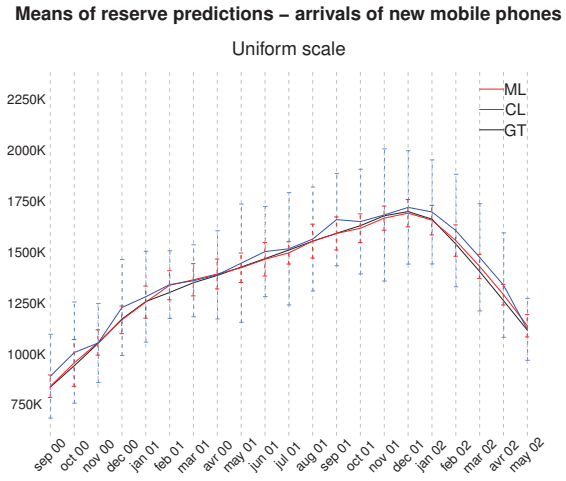

Fig. 3.19 Means of reserve predictions for the ML and CL method - arrival of new phone models (the vertical dashed segments provide the 95% prediction spans for each prediction)

3.5.5.4 Arrival of new phone models

In this scenario, we decided to change our portfolio structure by letting each brand introduce its Model 3 successively from October 2016 to December 2017. Brand 1 will release its model 3 at the end of October 2016, Brand 2 at the end of November 2016, etc... The distribution of the model types before (resp. after) the release of Model 3 is given in Table 6.

Moreover, it has been assumed that Model 3 of Brand 1 has a claim hazard rate twice as much as the other Model 3 of Brands 2, 3 and 4. Remind moreover that theft damage depends on phone model, so by introducing new phone models over time, we increase the global claim rate of our portfolio. The assumption of a stationary portfolio structure is no more satisfied, and this could introduce a bias in the CL reserves. Figure 19 shows that both methods actually handle correctly this change of structure in the portfolio (we also see the same differences between the CL and ML reserves variances):

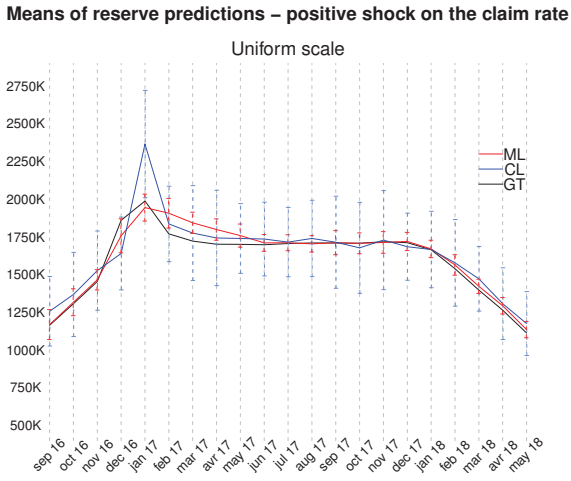

Fig. 3.20 Means of reserve predictions for the ML and CL methods - temporary claim rate shock (the vertical dashed segments provide the 95% prediction spans for each prediction)

Model type	Prob dist before	Prob dist after
0	0.15	0.05
1	0.35	0.10
2	0.50	0.35
3	0	0.50

Table 6: Model type distributions

3.5.5.5 Temporary claim rate shock

In this scenario, we set a raise of 40% on the claim hazard rate between mid-December 2016 and mid-January 2017. Then the hazard rate just comes back to its normal value. The assumption of a stationary claim rate and of a stationary occurrence delay are no more satisfied and this could make wrong predictions with the CL method.

We take from Figure 20 three comments. First, both methods under-estimate reserves for the reserving month of December 2016. Second, for the reserving month of January 2017, the CL method over-estimates the reserves whereas ML method is correct. Third, between February 2017 and June 2017, reserves are over-estimated with the ML method whereas the CL method is back to normal.

We explain these observations in the following way:

1. The CL reserves are computed using the 'stationary' development factor for reserving month December 2016. This factor is indeed under-estimated because we set a sharp raise of hazard rate between mid-December 2016 and the end of December 2016. This leads to an under-estimation of claim reserves in December 16 for January 2017, and this error is then propagated although correct development factors have been used.

   The ML reserves are computed by learning the claim rate observed in December for reserving month December 2016. Hence, the algorithm learns that there is a sharp raise in mid-December. But those additional claims which are happening in December are, for many of them at the end of December, declared in January. This explains the fact that the ML method is able to learn that there is a hazard rate rise, but the fact that those claims are not known yet by the insurer leads to a slight under-estimation of the real reserve (IBNR).

2. The CL reserves are computed using the new development factor, which takes into account the claim rate rise of December 2016. This development factor is over-estimated compared to the GT because we know that the claim rate rise is over by the end of January. Thus, we observe in January the additional claims which occurred in the end of December and in January. Hence, the reserves for the first month are computed using an over-estimated development factor, and on a higher amount of paid claims. This leads to a wide over estimation of reserves.

   The ML reserves are computed by learning the claim rate observed in January, and since it as the same as December, the estimation is just fine.

3. The CL reserves are computed using an over-estimated development factor, but this bias tends to zero over time since the claim rate is back to normal since mid-January 2017.

   The ML reserves are computed using one sub-model (i.e. $k = 1$). Thus, the large claim rate of December and January is learnt several times to compute long-horizon reserves over time. This leads to a bias in reserves calculation, which tends to zero over time, but slower than the CL method. This behavior can be overcome by using more than one sub-model (i.e. $k > 1$), because the bias is time-punctual. This means that we will have a few number of biased sub-models, and averaging the predictions of each sub-model would vanish the bias when k grows.

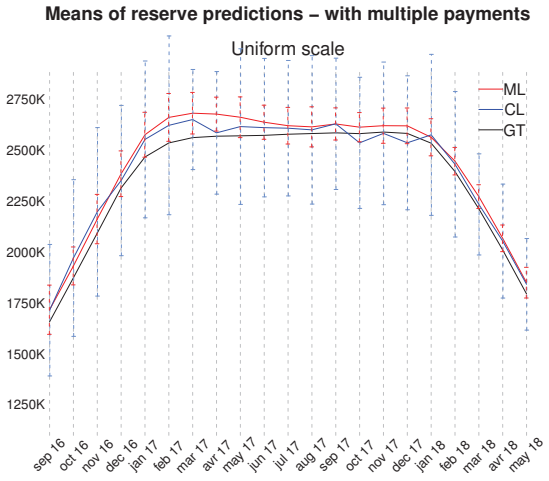

Fig. 3.21 Means of reserve predictions for the ML and CL methods - multiple payments (the vertical dashed segments provide the 95% prediction spans for each prediction)

This scenario illustrates a limit of our method, when we use only one sub-model to compute the ML reserves.

3.5.5.6 Multiple payments

In this scenario, we decided to modify the central scenario by allowing multiple payments although it is rarely the case for such a product. We permit the insurance company to pay the loss through four payments (the loss has been uniformly spread over the payments) and we assumed that the mean loss inter payment delay is 30 days. As a consequence, the IBNR reserves are not modified and the CL assumptions still hold. Only the RBNS reserves increase due to the multiple payments (the reserve are actually in general multiplied by 2). On the following figure, we depict the means and the 95% confidence intervals of the reserve predictions for the ML and CL methods.

We observe that the CL reserve predictions behave as in the one-payment case (the 95% confidence intervals are slightly larger). The ML reserve predictions have however a positive bias. To understand this bias, one has to look at the RBNS and IBNR reserves. Despite the additional uncertainty brought by the multiple payment dates and multiple payment amounts, the RBNS reserves are still well estimated by the ML algorithm without bias. However the IBNR reserves now include a bias although nothing has been changed with respect to the reporting delay. The main reason is that our IBNR reserve estimation is based on several frequency-loss models for each payment

and not on a unique frequency-loss model for the sum of the multiple payments in the previous case. The differences between the sizes of train and test databases make very small individual bias on the frequency predictions, which, when they are added in the final prediction, lead to a bias. Nevertheless the accurateness of the ML reserve predictions is still far better than the CL reserve predictions.

3.5.6 Discussion about individual/collective approaches

We present and discuss in this section prediction methods that we can classify between the CL method and the ML method. The CL method can be viewed as a collective model while the ML method can be viewed as an individual model with covariates. We therefore decided to consider an individual model without covariates relating to the policy or the policyholder (we call it an individual model without prior information) and an individual model where the "prior information" included in the covariates is not informative enough (we call it an individual model with no predictive power covariates).

Individual model without prior information

We considered the same model as in the central scenario, but we imposed to the ML algorithm not to use the policy-related features, which are the phone brand, the phone price, the phone model type, the coverage type ("Breakage", "Breakage and Oxidation", "Breakage and Oxidation and Theft"), and the following claim-related feature: the type of damage ("Breakage", "Oxidation", "Theft"). But we kept the date related features such as the underwriting date, the reporting delay, the number of days since the claim has been declared ..., because these features can be used anyway by all the insurance companies.

The results concerning the means of the reserve predictions and their 95% confidence intervals are depicted in the following figure.

It can be seen that the ML approach is not necessarily better than the CL approach as long as the size of the portfolio does not become constant around January 2017. The main reason is that the IBNR reserves are actually very poorly evaluated by the ML algorithm when it can not use the policy-related features and therefore there is no significant differences between ML and CL approaches for the global reserves. Nevertheless after January 2017, the ML algorithm improves the accurateness of the IBNR reserves (by stabilizing its predictions) and the individual approach (without prior information) becomes competitive even if the policy-related features are not used. Therefore the other features still bring a small but useful signal.

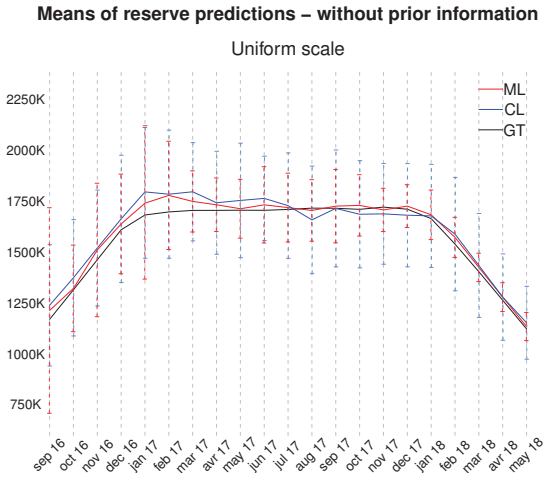

Fig. 3.22 Means of reserve predictions for the ML and CL methods - without prior information (the vertical dashed segments provide the 95% prediction spans for each prediction)

We assume that claim frequencies are generated through a competing model between risks, with constant incidence hazard rates that can depend on the mobile phone model type (but neither on the brand nor the model type), in the following way

Individual model with no predictive power covariates

We decided to modify the model of the central scenario by making the more informative covariates uninformative. More specifically:

- the prices of all mobile phones (for each brand and each mobile type) have been fixed to 500;

- the yearly incidence rate for each coverage types and for each model type has been fixed to 0.05;

- the parameters of the Beta distributions that characterize the distribution of the percentage of the price of the mobile phone to pay have been fixed to the parameters of Oxidation, i.e. 5 and 3 for the three coverage types (Breakage, Oxidation, Theft).

The results concerning the means of the reserve predictions and their 95% confidence intervals are depicted in Figure 23.

As for the previous model, we observe that ML approach is not necessarily better than the CL approach. It is quite natural to get such a result since the ML algorithms cannot use the uninformative covariates to improve the individual predictions.

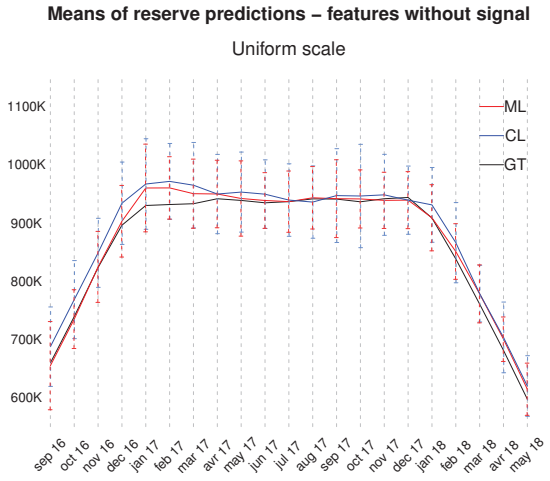

Fig. 3.23 Means of reserve predictions for the ML and CL methods - features without signal (the vertical dashed segments provide the 95% prediction spans for each prediction)

3.6 A real data set example

We illustrate our methodology with a Dutch loan insurance portfolio (provided by BNP Paribas Cardif). A first data set contains variables about 875 912 policyholders who underwrote an insurance cover from death, temporary or permanent disability, or unemployment. The underwriting period begins in 1998 and ends in 2018. A second data set contains 9581 claims, and a third data set records 128 890 payments (payment amount and payment date) for these claims.

The available variables characterizing the insurance policies and the policyholders are listed in Table 1. The insurance portfolio gathers 71 different types of contracts, distributed on 9 main networks, over 290 agencies, for a total of 10 124 brokers, implying a strong heterogeneity between policies.

% ANNUITY DECREASING	GENDER
% CHOSEN COMMISSION YEAR 1	SAMPLE AUTHORIZATION
% CHOSEN COMMISSION YEAR 2	INSURED AMOUNT / CAPITAL
% CHOSEN COMMISSION YEAR 3	INSURED AMOUNT TYPE
% PREMIUM INCREASE	INSURED ID
AGENCY NR.	ENTRY DATE IN THE SYSTEM
DISABILITY COVERAGE TYPE	LAST COVERAGE STATUS CHANGE
BIRTH DATE	LAST PAID BILLING DATE
BROKER NR.	NETWORK NR.
CERTIFICATE NR.	PROFESSION
CHILD ADDITION	PROFESSION CLASS
CLAIM PAYMENT DURATION	OCCUPATION CODE
CLAIM WAITING PERIOD	PREMIUM DURATION IN YEARS
COMMISSION SCHEMA	BANK (who borrows the mortgage amount)
COMMISSION TYPE	PAYMENT RECEIVED
CONTRACT TYPE	POLICY NR.
COVERAGE STATUS	PREMIUM INCREASE
COVERAGE TYPE	PREMIUM SCHEMA
COVERED JOB (ONLY FOR DISABILITY)	PREMIUM TYPE
COVERAGE END REASON	PRODUCT CODE
CURRENCY	PROFESSIONAL STATUS
DEATH COVERAGE TYPE	REPATRIATION
DISCOUNT AMOUNT	REVOLVING COMMISSION
DISCOUNT TYPE	SINGLE COMMISSION
DURATION IN MONTHS	SMOKING
EFFECT DATE	SURRENDERDATE
FISCALITY	TARIFF

Table 1: Overview of the available variables

Figure 24 illustrates how the portfolio size and the underwriting rate vary over time. It can be seen that, after a first period with a strongly increasing underwriting rate, it has been decided by the insurance company to lower the number of new policies from 2005, and, as a consequence, the portfolio will probably start its run-off period by the end of 2018.

For this case study, we decided to only focus on the coverages from disability and unemployment because these risks are the most difficult to predict since they heavily

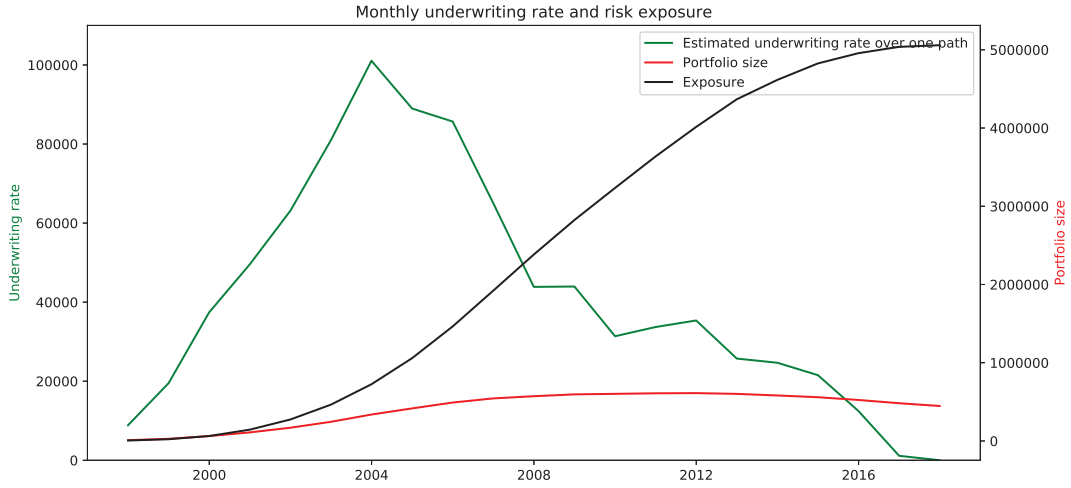

Fig. 3.24 Underwriting rate and risk exposure from 1998 to 2018

rely on the economical environment, although policyholders' characteristics (such as the professional status, the smoking habits, the age,...) are also important risk factors.

Figure 25 provides the time series of the monthly claim rates for both coverages. The unemployment claim rate is quite stable overtime, while the disability claim rate is multiplied by 6 over the ten year period, providing a possible explanation of the withdrawal of the insurance company from this market. Figure 26 visualizes the reporting delay over time. This time, the unemployment reporting delay is more stable than the disability reporting delay which significantly decreases from 2008. An explanation could come from changes of the claim waiting periods included in policies, as well as the economic context (beginning of the subprime crisis).

When considering claims, it is observed that payment durations can be very long (from a few months to several years). As a consequence there are many censored durations in the payment data set and it is not possible to compare predictions with respect to the true values (what we called previously the Ground Truth), even for the first underwriting years.

Therefore we decided to create artificial insurance policies where the payment duration is bounded by two years. In this way the true values of the payment durations are quickly observed and can be compared to predictions of models. Figure 27 illustrates

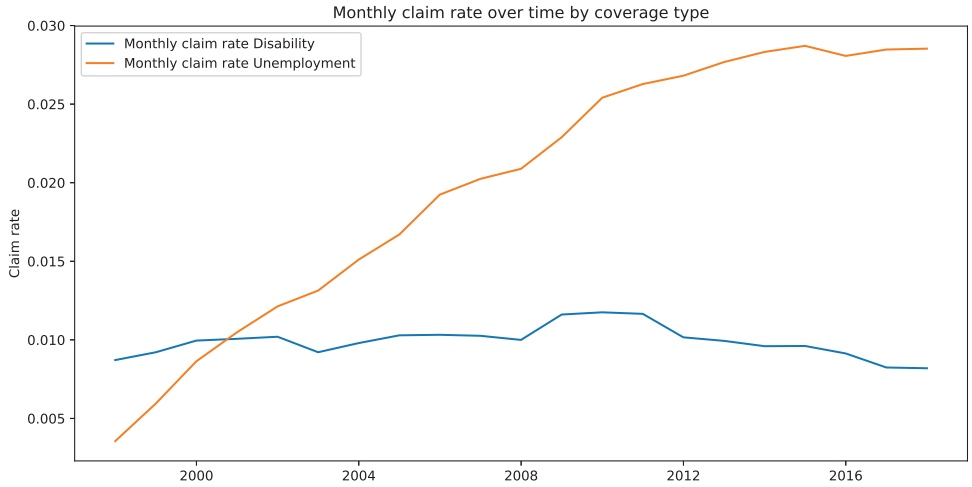

Fig. 3.25 Monthly claim rates for disability and unemployment from 1998 to 2018

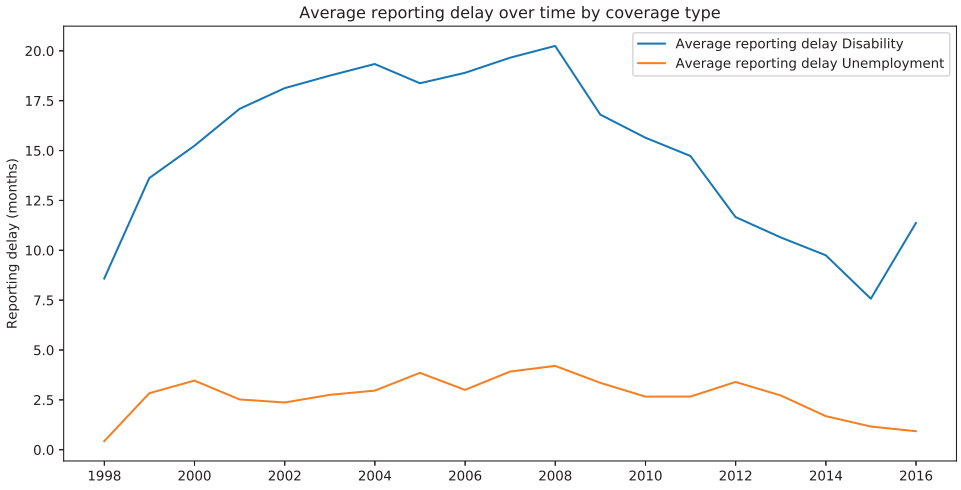

Fig. 3.26 Average reporting delays for disability and unemployment from 1998 to 2018

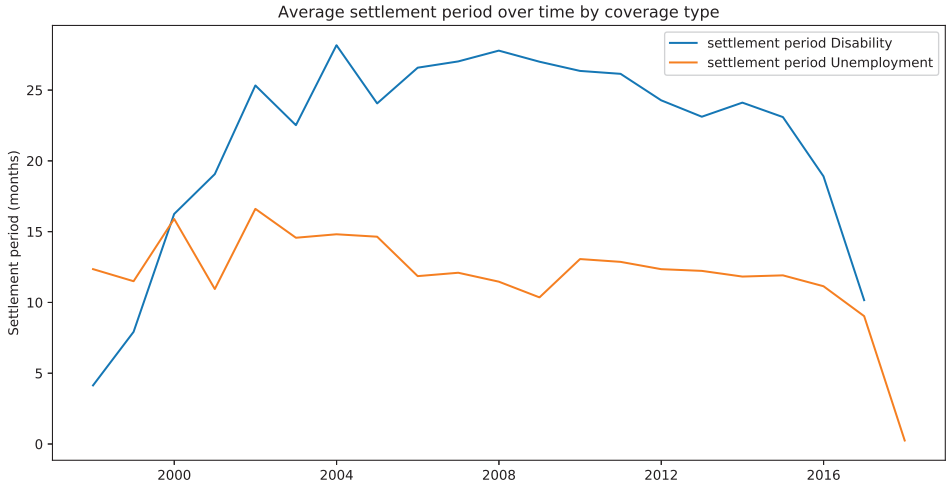

Fig. 3.27 Average settlement periods for disability and unemployment from 1998 to 2018

the average settlement periods (from the declaration date) for these artificial policies. The unemployment settlement period is roughly constant over time, while the disability settlement period increased a lot during the first 4 years of the portfolio.

The issue of censored data leads to difficulties to compare loss reserving methods. The right way to fairly do such an exercise is to wait until the observations are no more censored. In practice it is of course impossible so actuaries use statistics that take into account censoring to compare data and models. Here we decided to create artificial insurance policies.

We now use our approach for computing individual and aggregate claims reserves and compare them to the CL predictions. Before using the ExtraTrees algorithm, we build additional explanatory variables from the initial variables given in Table 1, and we create new variables describing claims (in particular in case of multiple and successive claims). We provide claim reserve predictions (overall, IBNR, RBNS) from 2004 to 2010, with their 95% confidence intervals, on respectively Figures 28, 29 and 30.

The 95% confidence intervals for the ML predictions are computed by bootstrapping the data and the 95% confidence intervals for the CL predictions use the standard errors based on Mack's formulas (Mack (1999)) with the Gaussian assumption. The

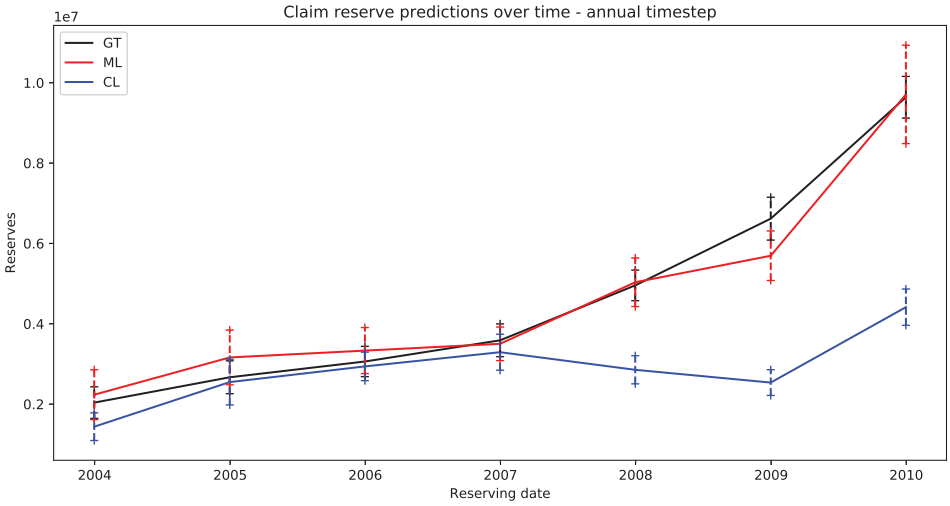

Fig. 3.28 Overall claim reserve predictions from 2004 to 2010

intervals for the GT are also computed by bootstrapping the data and are only used to provide an idea of the variability of the individual claim amounts. We used a traditional bootstrap procedure and considered random sampling with replacement. Bootstrapping indeed allows estimation of the sampling distribution of almost any unconditional statistic. But note that this type of bootstrapping can not be used to estimate conditional statistics, and particular procedure have to be implemented.

The ML predictions and CL predictions are quite equivalent from 2004 to 2007. However only the ML algorithm is able to take into account structural changes of the loan insurance portfolio after 2007. Note that the CL method strongly underestimates the true amounts of the reserves.

Figures 29 and 30 show that our approach leads respectively to very good IBNR and RBNS claim reserve predictions.

3.7 Summary, discussion and conclusion

We have proposed a new non-parametric approach for individual claims reserving. Our model is fully flexible and allows to consider (almost) any kind of feature information. As a result we obtain IBNR and RBNS claims reserves for individual policies integrating all available relevant feature information.

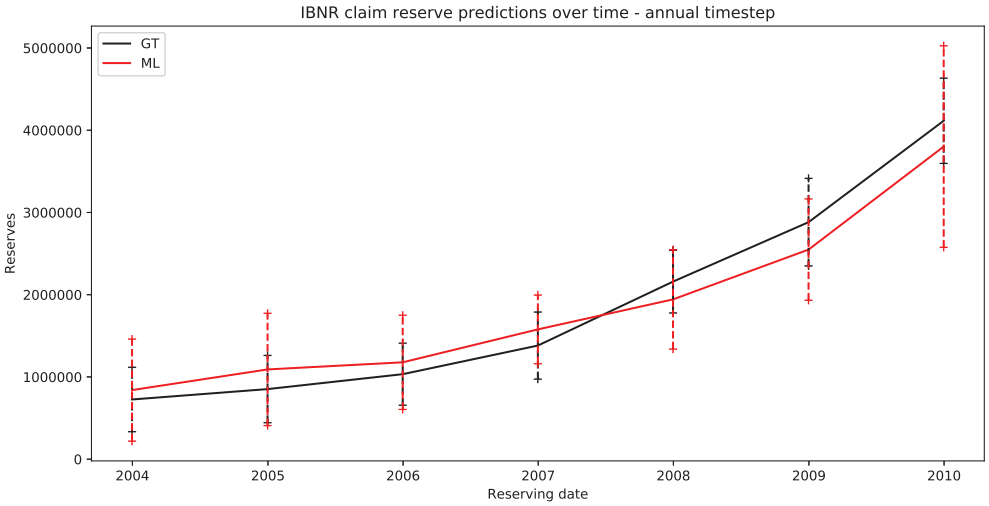

Fig. 3.29 IBNR reserve predictions from 2004 to 2010

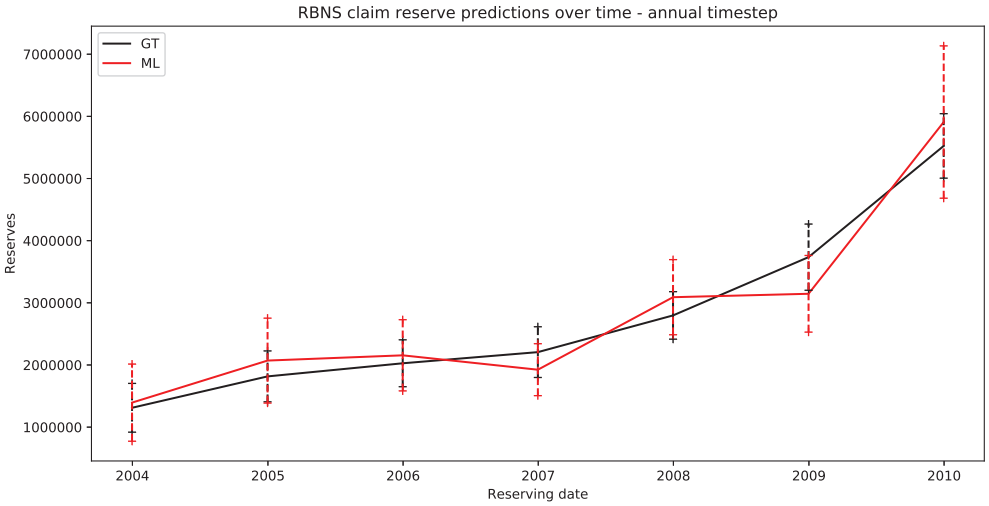

Fig. 3.30 RBNS reserve predictions from 2004 to 2010

In the Introduction, we have identified several issue concerning the CL method that we can now compare to our approach.

- First, we have underlined that there could be an over parameterization risk induced by a large number of tail factors that have to be estimated as compared to the number of components of the run-off triangles. Since the CL method works with aggregated data, it is clear that the number of parameters of the model is large with respect to the number of observations. In our ML method, we used all the individual data, but we rather decided to favor a non-parametric procedure. Therefore the comparison is not evident on this point.

- Second, we have explained that there could be a risk of error propagation through the development factors associated to a possible huge estimation error for the latest development periods. Our ML method is additive while the CL method is multiplicative. Therefore the risk of propagation of error disappears with our approach.

- Third, we have identified that there could be a potential lack of robustness and the need for appropriate treatments of outliers. To compare the CL and ML methods, we have to know whether the value of the outliers is the consequence of extreme values or the consequence of some (e.g. operational) changes. In the first case aggregated methods as CL method are in general more robust while in the second case, our individual method will perform better.

- Fourth, we have said that it is impossible to separate the assessments of IBNR and RBNS reserves. Our ML method clearly proposes a way to separate IBNR from RBNS.

In our case study where we used a Machine Learning algorithm known as ExtraTrees algorithm, we observe that the method provides almost unbiased estimators of the claims reserves with very small standard deviations (actually four to five times smaller than the Mack Chain Ladder standard deviation based on aggregate data). Moreover the ML estimators are more responsive to any changes in the development patterns of claims including occurrence, reporting, cost modifications,... than the CL estimators based on aggregate loss data. This is also what we have observed in our real data example.

We however have to shade these statements in light of the durations of the liabilities of the insurance company. It is well-known that the Chain Ladder method tends to perform better for short-tailed lines than for long-tailed lines. The reason is that any accident year reaches final settlement in a relatively short period of time. For long-tailed lines, the CL method has fewer and fewer points from which to calculate the individual development factor as long as the computation is done in the tail. Actually,

it is needed to decide at which development period to incorporate a tail factor that estimates development beyond the last observations. There are several techniques to estimate a tail such as using an implied industry-wide tail factor (if available), using an inverse power or Weibull curve-fitting technique, or estimating the decay in the selected development pattern and extending that decay assumption into future development periods. However the actuary must decide when to incorporate other methods from which to estimate ultimate losses. For long-tailed lines, our algorithm will also suffer from the small number of data when predicting loss reserves for policies with very long payment delays. For these policies, the depths of the trees used by the ExtraTree algorithm will be very small and the predictions will be close to the empirical means computed on the train sets. It is therefore clear that the ML algorithm would also benefit from experts' opinions.

Besides, other more complex case studies using unstructured information like texts or images could be considered. Text mining algorithms or Deep Convolution Neural networks could actually be then used to build informative features that would be incorporated in the process $(I_t)_{t\geq T_2}$ that gathers information on the individual claim history. As we experienced, such features can be very useful. However these more complex case studies will not change the main conclusion of the discussed case study of the paper, which is that our approach can provide for some insutance portfolios more efficient estimators of the claims reserves than the oldest aggregated data methods.

Acknowledgments

We would like to thank the actuarial department of BNP Paribas Cardif for fruitful discussions about the case study, Philippe Baudier, Pierre de Sahb and Sebastien Conort from the Data Lab of BNP Paribas Cardif for a lot of important discussions about the methodology, its implementation (in particular the way to build the train and test sets) and for checking a large part of our computer code.

We also thank the two anonymous reviewers for their thorough reviews and we highly appreciated the comments and suggestions, which significantly contributed to improving the quality of the paper.

Chapter 4

RadialStyle: an efficient Algorithm to Leverage multiple datasets for predictive modelling

4.1 Introduction

Machine Learning models usually require a large amount of data to be efficient. This is caused by the complexity of the phenomenon to be modeled and/or by the fact that the model is complex by nature.
Such an amount of data may not be always available in a single dataset, which can be caused by several reasons. For example, mergers and acquisitions for companies lead to having multiple datasets (from multiple companies) tackling the same prediction problem. On the other hand, clinical studies usually have a small amount of data available in a single dataset, but many laboratories carry out the same study, which motivates the use of a framework to share the predictive information lying in each study.
The use of a framework to fuse the multiple sources of information allows to leverage the predictive power from multiple datasets since it takes into account the heterogeneous behavior of sub-populations from one dataset to another.

Training a prediction model from multiple datasets raises two main challenges: **feature mismatch**, and **distribution mismatch**.
We define the feature mismatch as the fact that two datasets may not share exactly the same attributes.

We define the distribution mismatch as the fact that the common attributes from one dataset to another are not necessarily identically distributed. For example, the proportion of male and female from one dataset to another is not necessarily the same. One simple way (yet efficient) to combine distinct datasets is to concatenate them and force each missing value (due to feature mismatch) to be zero (or another relevant value). However, this method performs poorly when the feature mismatch proportion is high.

An efficient way to combine multiple datasets is to transport the distribution of one dataset to another. This allows to perform an efficient data augmentation by transforming one dataset to another. Hence, it opens the way to training a prediction model on a larger number of rows while taking into account i) the feature and distribution mismatch and ii) the predictive power of each dataset. Such a task is refered as the *domain translation* area, which is a direct application of optimal transport (see [Villani, 2009, Peyré et al., 2019]). Deep translation models have been largely studied in image translation [Isola et al., 2016, Zhu et al., 2017b, Zhu et al., 2017a, Choi et al., 2018, Kim et al., 2017], as well as in Natural Language Processing (NLP) [Collobert and Weston, 2008, Sutskever et al., 2011, Socher et al., 2010, McCann et al., 2017, Lample et al., 2018, Lample and Conneau, 2019], but very few has been done on tabular data.
Another issue that occurs when we want to transform one dataset to another is the quadratical number of transfer models to train. For example, if f is a model trained to transform exactly one dataset to another, and M is the number of available datasets, then we need to train $M(M-1)$ distinct models f to transform any of the M datasets to another one.
[Yoon et al., 2018b] addresses the problem of feature and distribution mismatch as well as the quadratical number of transfer models to train by training an ensemble of adversarial *encoders* and *decoders* to project each dataset in a latent central space and decode it back to the original domain.

RadialGAN rely on Generative Adversarial Neworks (GANs) [Goodfellow et al., 2014], which are known to be hard to train and can have some instability issues (e.g. mode collapse) [Salimans et al., 2016]. This motivates us to propose a transfer method for tabular data without the need to use a GAN framework. For that matter, our proposed method is inpired by the Style Transfer literature.

Style Transfer networks [Gatys et al., 2015a] have received a lot of attention, particularly with the recent improvements due to the use of layers such as Feature-wise Linear Modulation (FiLM) [Perez et al., 2018] and Adaptive Instance Normalization (AdaIN) [Huang and Belongie, 2017]. [Dumoulin et al., 2016] propose a network which outputs its *content input* picture, but in the same style than the *style input* picture by introducing conditional normalization layers. On the other hand, [Ghiasi et al., 2017] improves the same task by using FiLM blocks in its architecture. More generally, feature-wise transformations are applied in a broad range of deep learning applications detailed in [Dumoulin et al., 2018], and allows to process efficiently multiple heterogeneous inputs.

To the best of our knowledge, Style Transfer models have been applied to image processing or equivalently video processing [Yang et al., 2018], but not on tabular data. This is due to the fact that the Style Transfer losses are designed to work on images. In this paper, we propose a Style Transfer framework, which we call RadialStyle, to transform one dataset to another, in order to perform a data augmentation without losing the content of a dataset (i.e. the predictive signal), allowing to leverage the predictive signal of multiple distinct datasets. To do so, we define an appropriate content and style losses for the Style Transfer network. We address the problem of feature and distribution mismatch by jointly training an ensemble of *auto-encoders*, then performing the style transfer directly through a latent space.

We illustrate our method with several experiments built from three real-world datasets (which have distinct properties), and show that it outperforms the RadialGAN algorithm and give better results than the trivial benchmark. We also change the experimental setup to study the behavior of RadialGAN and RadialStyle, and discuss the results. Section 4.3 and 4.4 give more details about the Style Transfer networks and the RadialGAN algorithm (coupled with related work), Section 4.5 gives details about the RadialStyle and Section 4.6 illustrates the method and discusses the results.

4.2 Related Work

The problem of leveraging multiple datasets to impove a prediction task has already been addressed in different settings. This section situates our approach with existing methods

Multi-Source Learning

[Heskes, 2000, Evgeniou and Pontil, 2004, Liao et al., 2005] successfully address the problem of distribution mismatch but assume that each dataset share exactly the same attributes, hence those approaches are not appropriate in presence of feature mismatch. [Wiens et al., 2014] address the problem of feature mismatch, but does not take into account the distribution mismatch between two distinct datasets.

Domain Translation

Domain translation models are neural networks which are trained to transport an observed distribution to another target distribution. Image-to-Image translation models such as Pix2Pix [Isola et al., 2016], CycleGAN [Zhu et al., 2017a] or DiscoGAN [Kim et al., 2017] focus on the task of transforming one characteristic of an image to another. Those models address both feature and distribution mismatch since they are trained on unpaired heterogeneous images. However, these methods only perform a *pairwise* domain translation, which means that they are not designed to address the multi-domain issue (which leads to a quadratical number of transfer models to train).

Multi-Domain Translation

StarGAN [Choi et al., 2018] address the problem of multi-domain translation by mapping all datasets to the target domain at once (using a single generator), but the latter only works when there is no feature mismatch.
RadialGAN [Yoon et al., 2018b] address the problem of multi-domain translation as well as the feature and distribution mismatch by jointly training an ensemble of encoders and decoders to i) map all datasets to a latent space and ii) map back to the original feature spaces. However, such a method does not take into account the specificity of the target domain that the other datasets have to be mapped to. This leads in some cases to unexpected underperformance due to the fact that each encoder/decoder is forced to be domain agnostic.

4.3 Style Transfer

Convolutional Neural Networks (CNN) [LeCun et al., 1989] are composed of layers of small computational units (kernels) that slide through the data and process visual information hierarchically in a feed-forward framework. Each layer of units can be seen as a collection of image filters, each of which extracts a certain feature from the input image. The output of such a layer is called *feature map*, which contains differently

filtered versions of the input image.

In computer vision, a lot of work has been done in training computers to capture the artistic style of a painting in order to reproduce this style in an arbitrary picture. Advances on image texture representation have shown that deep convolutional networks are able to estimate intermediate representations of the image, and such representations contain more and more details of the image as the depth of the layer increases [Gatys et al., 2015b, Mahendran and Vedaldi, 2015]. Therefore, through the convolutions of the network, the input image is transformed into representations that increasingly care about the details of the content image. The information contained in each layer about the input image can be directly visualised by constructing the image from the feature map of that layer.
Higher layers in the network capture the macro-level content in terms of objects and their arrangement in the input image without taking care about the exact values of each pixel in the reconstruction. In contrast, lower layers reproduce the exact pixel values of the input image.

[Gatys et al., 2015a] show that the *style* and the *content* of a painting are separable by proposing a method to reproduce an input *content* image in a given painting *style*, based on the intermediate representation of the state-of-the-art Deep Convolutional Neural Network (DCNN) trained on image recognition [Simonyan and Zisserman, 2014]. This enlightens a new family of neural networks: Style Transfer nets.

The total loss $\mathcal{L}_{total}$ of such networks is decomposed in *content* and *style* losses $\mathcal{L}_c$ and $\mathcal{L}_s$ respectively.
The underlying philosophy of such losses is the following:

- Two images are similar in content if their high-level feature maps as extracted by an image recognition system are close in Euclidean distance.
- Two images are similar in style if their low-level features as extracted by an image recognition system share the same spatial statistics.

The second definition measures the spatial statistics distance by computing the correlations of the activations of the lower feature maps through the underlying Gram matrices of such feature maps. We denote by $\mathcal{G}[.]$ the gram matrix associated with a given feature map. One may either use a known Deep CNN (DCNN) architecture like the VGG-Network [Simonyan and Zisserman, 2014] or the *inception network*

[Szegedy et al., 2015, Szegedy et al., 2016, Szegedy et al., 2017] as the image recognition system, or train a CNN from scratch from ImageNet for example.

More formally, for a content image c and a style image s, we can write the optimization problem for a generated image x as the following:

$$\min_x \mathcal{L}_{total}(x, c, s) = \min_x \lambda_c \mathcal{L}_c(x, c) + \lambda_s \mathcal{L}_s(x, s), \tag{4.3.1}$$

where $\lambda_c, \lambda_s > 0$ are hyper-parameters. Usually, we fix $\lambda_c = 1$ and λ_s is seen as a Lagrange multiplier weighting the relative strength of the style loss.
We denote by $\mathcal{C}$ (resp. $\mathcal{S}$) the higher (resp. lower) level layers of an image classification system. Thus, we can write the content and style losses as follows:

$$\mathcal{L}_c(x, c) = \sum_{j \in \mathcal{C}} \frac{1}{n_j} ||f_j(x) - f_j(c)||^2_{L^2}, \tag{4.3.2}$$

$$\mathcal{L}_s(x, s) = \sum_{i \in \mathcal{S}} \frac{1}{n_i} ||\mathcal{G}[f_i(x)] - \mathcal{G}[f_i(c)]||^2_F, \tag{4.3.3}$$

where f_i are the network activations at layer i, n_i is the number of units at layer i, and $||.||_F$ is the Frobenius norm.

Feature-wise transformations

Style transfer models, among many other models, have to deal with multiple heterogeneous inputs. The stake is to properly take into account such an heterogeneity in order to be able to generalize to multiple *contexts*. The problem of finding an effective way to fuse multiple sources of information is still open. One effective approach is to process *feature-wise transformations*, which consists of adding a block to a neural network which aims at modulating the information conditionally on a given context input.

[Perez et al., 2018] propose to use Feature-wise Linear Modulation (FiLM) blocks to apply a feature-wise affine transformation to the input $\mathbf{x}$ conditionally on an additional input $\mathbf{z}$:

$$FiLM(\mathbf{x}) := \alpha(\mathbf{z}) * \mathbf{x} + \beta(\mathbf{z}), \tag{4.3.4}$$

where α and β are $\mathbf{z}$-dependent **scaling** and **shifting** parameters respectively, and $*$ is the Hadamard (element-wise) product. The FiLM block predicts the scale and shift

parameters according to the conditional input $\mathbf{z}$, then the transformation is applied to $\mathbf{x}$.

[Huang and Belongie, 2017] propose an Adaptive Instance Normalization which does not have learnable affine parameters, but instead apply a rescaling of the input $\mathbf{x}$ to the scale and shift of the conditional input $\mathbf{z}$:

$$AdaIN(\mathbf{x}, \mathbf{z}) := \sigma(\mathbf{z}) \left(\frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \right) + \mu(\mathbf{z}). \tag{4.3.5}$$

4.4 Background

Let $X^{(1)}, \dots, X^{(M)}$ be M random variables taking values in $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(M)}$ respectively, where $\mathcal{X}^{(i)} \subset \mathbb{R}^{p_i}$, $i \in \{1, \dots, M\}$, and let Y be a random variable taking its values in a label space $\mathcal{Y}$, either continuous or discrete.
Let $\mathcal{D}_i := \{(x_j^{(i)}, y_j^i)\}_{j=1}^{n_i}$ be M datasets. $(x_j^{(i)}, y_j^i)$ are iid samples drawn from the joint distribution $(X^{(i)}, Y) \in \mathcal{X}^{(i)} \times \mathcal{Y}$.

We aim at building M predictors $f_i : \mathcal{X}^{(i)} \to \mathcal{Y}$ such that each predictor uses the predictive information of every other dataset $\mathcal{D}_j$, $j \neq i$. One way to take into account such information is to build M *augmented* datasets $\mathcal{D}_i' := \mathcal{D}_i \cup_{j \neq i} \widehat{\mathcal{D}}_{ji} \in \mathcal{X}^{(i)} \times \mathcal{Y}$, where $\widehat{\mathcal{D}}_{ji}$ are estimated transformation of $\mathcal{D}_j$ into $\mathcal{D}_i$ $(j \neq i)$, then train f_i on $\mathcal{D}_i'$.

4.4.1 RadialGAN

RadialGAN [Yoon et al., 2018b] uses an adversarial framework to simultaneously train M *encoders* and *decoders* F_i, G_i, such that $F_i : \mathcal{X}^{(i)} \times \mathcal{Y} \to \mathcal{Z}$ and $G_i : \mathcal{Z} \to \mathcal{X}^{(i)} \times \mathcal{Y}$, where $\mathcal{Z}$ is a latent space in which all the datasets are *projected*.
The i-th adversarial loss for such a model is defined by:

$$\mathcal{L}_{adv}^i(D_i, G_i, \{F_j; j \neq i\}) = \mathbb{E}[\log D_i(X^{(i)}, Y)] + \sum_{j \neq i} \alpha_{ij} \mathbb{E}[\log(1 - D_i(G_i(F_j(X^{(j)}, Y))))], \tag{4.4.1}$$

where $\alpha_{ij} = \frac{n_j}{\sum_{k \neq i} n_k}$ and $D_i : \mathcal{X}^{(i)} \times \mathcal{Y} \to [0, 1]$ is the discriminator of the i-th domain, which attempts to distinguish between the real samples (drawn from $(X^{(i)}, Y)$) from the fake samples (drawn from $(\widehat{X}^{(i)}, \widehat{Y}) = G_i(F_j(X^{(j)}, Y))$). The i-th *cycle-consistency* loss is defined by:

$$\begin{aligned}\mathcal{L}^i_{cyc}(G_i, \mathbf{F}) = \; & \mathbb{E}[||(X^{(i)}, Y) - G_i(F_i(X^{(i)}, Y))||_2] \\ & + \textstyle\sum_{j \neq i} \alpha_{ij} \mathbb{E}[||F_j(X^{(j)}, Y) - F_i(G_i(F_j(X^{(j)}, Y)))||_2],\end{aligned} \tag{4.4.2}$$

where $\mathbf{F} = (F_1, \ldots, F_M)$ and $||.||_2$ is the L_2 norm.

The *cycle-consistency* loss ensures that encoding to the latent space then decoding gives a result close to the original input. In other words, we expect that $G_i(F_i(X^{(i)}, Y)) \approx (X^{(i)}, Y)$ and $F_i(G_i(Z)) \approx Z$.

The global objective function for the RadialGAN framework is defined, using the losses $\mathcal{L}_{adv}$ and $\mathcal{L}_{cyc}$, as the following minimax problem:

$$\min_{\mathbf{G},\mathbf{F}} \max_{\mathbf{D}} \left(\sum_{i=1}^{M} \mathcal{L}^i_{adv}(D_i, G_i, \{F_j; j \neq i\}) + \lambda \sum_{i=1}^{M} \mathcal{L}^i_{cyc}(G_i, \mathbf{F}) \right), \tag{4.4.3}$$

where $\mathbf{F} = (F_1, \ldots, F_M)$, $\mathbf{G} = (G_1, \ldots, G_M)$, $\mathbf{D} = (D_1, \ldots, D_M)$, and λ is a hyper-parameter.

The training of the RadialGAN is done by alternatively training $\mathbf{D}$ with fixed $\mathbf{F}$ and $\mathbf{G}$, then training $\mathbf{F}$ and $\mathbf{G}$ with fixed $\mathbf{D}$ until an equilibrium between the loss of $\mathbf{D}$ and the loss of $\mathbf{F}$ and $\mathbf{G}$ is found.

4.5 Model

We start from the notations defined in Section 4.4. Recall that we aim at training M predictors $f_i : \mathcal{X}^{(i)} \to \mathcal{Y}$, $i \in \{1, \ldots, M\}$, and leverage the predictive information contained in every other dataset $\mathcal{D}_j$ $(j \neq i)$ by performing a data augmentation with an estimated transformation of each dataset $\mathcal{D}_j$ to a dataset close to $\mathcal{D}_i$, denoted $\widehat{\mathcal{D}}_{ji}$ (dataset transferred from the j-th domain to the i-th domain).

We first train M *encoders* and *decoders* F_i, G_i, such that each dataset $\mathcal{D}_i$ can be encoded to a latent space $\mathcal{Z}$ and decoded from $\mathcal{Z}$ to $\widehat{\mathcal{D}}_i \approx \mathcal{D}_i$. We denote by $\mathcal{D}_i^{\mathcal{Z}} := F_i(\mathcal{D}_i)$ the projection of $\mathcal{D}_i$ in $\mathcal{Z}$. The encoders and decoders aim at disentangling the heterogeneity between each feature space $\mathcal{X}^{(i)}$, since such spaces may not share the same dimension and the random variables $X^{(i)}$ may not share the same distribution.
We then train a *style transfer* model S_i to perform a (latent) *domain transfer* directly

through $\mathcal{Z}$ by transferring from a latent domain $\mathcal{D}_j^{\mathcal{Z}}$ to $S_i(\mathcal{D}_j^{\mathcal{Z}}) := \widehat{\mathcal{D}}_{ji}^{\mathcal{Z}}$ which is close to $\mathcal{D}_i^{\mathcal{Z}}$.
S_i is trained to minimize the L_2 norm between $\mathcal{D}_i^{\mathcal{Z}}$ and $\widehat{\mathcal{D}}_{ji}^{\mathcal{Z}}$ and a loss $\mathcal{L}$ between Y and $\widehat{Y} := G_i^{p_i+1}(Z)$, where $G_i^{p_i+1}(.) \in \mathcal{Y}$ is the last element of the vector $G_i(.)$. The first loss ensures that we translate well through the latent space to the appropriate target domain, and the second loss ensures that we translate with a minimum loss of predictive information between $\mathcal{Z}$ and $\mathcal{Y}$.

For a dataset $\mathcal{D}_j$ to be translated into a target dataset $\mathcal{D}_i$, we perform the translation by the following steps:

1. We encode $\mathcal{D}_i$ (resp. $\mathcal{D}_j$) to the latent space $\mathcal{Z}$ to get $\mathcal{D}_i^{\mathcal{Z}} = F_i(\mathcal{D}_i)$ (resp. $\mathcal{D}_j^{\mathcal{Z}} = F_j(\mathcal{D}_j)$).
2. We translate $\mathcal{D}_j^{\mathcal{Z}}$ to $\widehat{\mathcal{D}}_{ji}^{\mathcal{Z}}$ through $\mathcal{Z}$ with a style transfer model S_i, such that $\widehat{\mathcal{D}}_{ji}^{\mathcal{Z}} = S_i(\mathcal{D}_i^{\mathcal{Z}}, \mathcal{D}_j^{\mathcal{Z}}) \approx \mathcal{D}_i^{\mathcal{Z}}$.
3. We then decode the estimated latent translation with G_i, i.e. $\widehat{\mathcal{D}}_{ji} = G_i(\widehat{\mathcal{D}}_{ji}^{\mathcal{Z}})$.

The global architecture of our approach is depicted in Figure 4.1. Figure 4.2 illustrates the domain transfer and how each elements of our structure are used.

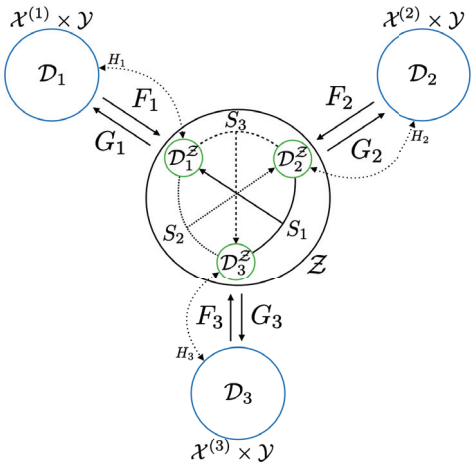

Fig. 4.1 RadialStyle Architecture. $\mathcal{X}^{(i)} \times \mathcal{Y}$: domain of dataset $\mathcal{D}_i$, $\mathcal{Z}$: latent space, F_i, G_i: encoder and decoder of the i-th dataset, D_i: i-th predictor from $\mathcal{Z}$ to $\mathcal{Y}$, $\mathcal{D}_i^{\mathcal{Z}}$: projection of $\mathcal{D}_i$ in $\mathcal{Z}$, S_i: style transfer model, transferring latent domains $\mathcal{D}_j^{\mathcal{Z}}$, $j \neq i$ to the target latent domain $\mathcal{D}_i^{\mathcal{Z}}$.

4.5.1 Encoders and Decoders

The first step of our method consists of encoding each dataset $\mathcal{D}_i$ to a latent space $\mathcal{Z}$. We do so to address the problem of dimension mismatch between $\mathcal{X}^{(i)}$ and $\mathcal{X}^{(j)}$ $(i \neq j)$ and distribution mismatch between $X^{(i)}$ and $X^{(j)}$.
Such a mismatch is caused by the fact that there is no reason for two distinct datasets to share the same amount of attributes nor the same distribution.

Let $F_i : \mathcal{X}^{(i)} \times \mathcal{Y} \to \mathcal{Z}$ and $G_i : \mathcal{Z} \to \mathcal{X}^{(i)} \times \mathcal{Y}$ be respectively M *encoders* and *decoders*. We jointly train F_i, G_i to satisfy the *auto-encoding* loss, i.e. the ability to encode to the latent space and decode back to the original domain with a minimum loss of information.
We denote by $\mathcal{D}_i^{\mathcal{Z}}$ the projection of the dataset $\mathcal{D}_i$ to $\mathcal{Z}$. Hence we have $\mathcal{D}_i^{\mathcal{Z}} = F_i(\mathcal{D}_i)$.

The auto-encoders are trained to minimize the L_2 distance between $\mathcal{D}_i$ and $\widehat{\mathcal{D}}_i := G_i(\mathcal{D}_i^{\mathcal{Z}})$. We define the objective of the auto-encoders with the following formula:

$$\min_{\mathbf{F},\mathbf{G}} \sum_{i=1}^{M} \mathbb{E}\left[||(\mathcal{D}_i - G_i(F_i(\mathcal{D}_i)))||_F\right] = \min_{\mathbf{F},\mathbf{G}} \sum_{i=1}^{M} \mathbb{E}\left[\left|\left|\Big((X^{(i)}, Y) - G_i(F_i(X^{(i)}, Y))\Big)\right|\right|_2^2\right], \tag{4.5.1}$$

where $\mathbf{F} = (F_1, \ldots, F_M)$ and $\mathbf{G} = (G_1, \ldots, G_M)$, $||.||_2$ is the L_2 norm and $||.||_F$ is the Frobenius norm.

The loss in Equation 4.5.1 is the first term of the *cycle-consistency* loss from the RadialGAN in Equation 4.4.2. Moreover, such auto-encoders are different from the RadialGAN's objective since we do not give an additional *adversarial* loss.
In our experiments, we implemented F_i and G_i as multi-layer perceptrons.

4.5.2 Transfer

The transfer step aims in training M Style Transfer models $S_i : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ such that $\widehat{\mathcal{D}}_{ji}^{\mathcal{Z}} := S_i(\mathcal{D}_i^{\mathcal{Z}}, \mathcal{D}_j^{\mathcal{Z}})$ is close to $\mathcal{D}_i^{\mathcal{Z}}$ $\forall j$ while preserving the predictive signal between $\mathcal{D}_j^{\mathcal{Z}}$ and the target of $\mathcal{D}_j$.
We define the first input of S_i as the *style* input, and the second input as the *content* input.

The Style Transfer model S_i is trained to minimize the style and content losses $\mathcal{L}_s^i$ and $\mathcal{L}_c^i$, which we define with the following formulas:

$$\mathcal{L}_s^i(S_i, \mathbf{F}) = \sum_{j \neq i} \mathbb{E}\left[\left|\left|S_i(F_i(X^{(i)}, Y), F_j(X^{(j)}, Y)) - F_i(X^{(i)}, Y)\right|\right|_2\right], \tag{4.5.2}$$

$$\mathcal{L}_c^i(S_i, G_i, \mathbf{F}) = \sum_{j \neq i} \mathbb{E}\left[\mathcal{L}(G_i^{p_i+1}(S_i(F_i(X^{(i)}, Y), F_j(X^{(j)}, Y))), Y)\right], \tag{4.5.3}$$

where $\mathcal{L}$ is an appropriate loss according to the nature of $\mathcal{Y}$, e.g. the *cross-entropy loss* if $\mathcal{Y}$ is discrete, and the L_2 loss if $\mathcal{Y}$ is continuous. The latter loss ensures that we transfer $\mathcal{D}_j^{\mathcal{Z}}$ to $\mathcal{D}_i^{\mathcal{Z}}$ without losing the predictive information of $\mathcal{D}_j^{\mathcal{Z}}$.

Using the losses $\mathcal{L}_s^i$ and $\mathcal{L}_c^i$ we define the objective of the RadialStyle as the following optimization problem:

$$\min_{S_i} \mathcal{L}_c^i(S_i, G_i, \mathbf{F}) + \lambda \mathcal{L}_s^i(S_i, \mathbf{F}) \quad \forall i \in \{1, \ldots, M\}, \tag{4.5.4}$$

where $\mathbf{F} = (F_1, \ldots, F_M)$, and $\lambda > 0$ is a hyperparameter.

The architecture of the Style Transfer model is detailed in Figure 4.3

4.6 Experiments

In this section, we illustrate and compare our method with RadialGAN. We were not able to get a set of multiple datasets tackling the same prediction problem. However, we reproduced such a setup (for the sake of the experiments) by making a random split of a single dataset (both on rows and columns) then by training the transfer model (RadialGAN or RadialStyle) from the multiple splits.

Note that in this particular setup (splitting a single dataset) we do not expect a predictor f_i' trained on an augmented dataset $\mathcal{D}_i'$ (built from multiple splits of a larger dataset) to give better performances than a predictor trained on the whole raw dataset. However, we expect it to outperform a predictor trained on a single split and to give better performances than the Simple-Combine benchmark, which we define in the next subsection.

We illustrate our approach through experiments built from several real-world datasets,

namely the *Census Income KDD* dataset [Dua and Graff, 2017], the *Facebook Comments* dataset [Kamaljot et al., 2015] and the *BNP Paribas Cardif's Claim Management* dataset[1] from the Kaggle challenge hosted in 2015.

4.6.1 Experimental setup

For a dataset $\mathcal{D}$, we sampled (without replacement) M sub-datasets $\mathcal{D}_1, \dots, \mathcal{D}_M$ from $\mathcal{D}$. Each sub-dataset $\mathcal{D}_i,\ i \in \{1, \dots, M\}$, contains p_i attributes (drawn at random) and n_i rows (also drawn at random).
We perform an additional sampling of $\mathcal{D}$ into M *holdouts* $\mathcal{D}_1^{\text{ho}}, \dots, \mathcal{D}_M^{\text{ho}}$, except that we keep the same p_i attributes as $\mathcal{D}_i$, and we do not include the same rows as $\mathcal{D}_i$ in the random choice of rows.

Once we splitted $\mathcal{D}$, we train the RadialGAN and RadialStyle models from $\mathcal{D}_1, \dots, \mathcal{D}_M$ and build M *augmented* datasets $\mathcal{D}'_{i,\text{G}} = \mathcal{D}_i \cup_{i \neq j} \widehat{\mathcal{D}}_{ji}$ following the RadialGAN algorithm and M augmented datasets $\mathcal{D}'_{i,\text{S}}$ following RadialStyle.
We then train a prediction model f_i on $\mathcal{D}_i$ and two other prediction models $f'_{i,\text{G}}$ and $f'_{i,\text{S}}$, trained on $\mathcal{D}'_{i,\text{G}}$ and $\mathcal{D}'_{i,\text{S}}$ respectively, and compare the performance gain between f_i and $f'_{i,\text{G}}$, and between f_i and $f'_{i,\text{S}}$.
We also compare $f'_{i,\text{G}}$ and $f'_{i,\text{S}}$ with a prediction model f_i^c trained on a simple combination of all datasets denoted $\mathcal{D}^c := \{(x_i, y_i)\}_{i=1}^n$, where $n = \sum_i n_i$ and (x_i, y_i) are iid samples drawn from $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} := \cup_i \mathcal{X}^{(i)}$. In other words, we compare each augmentation method against the *simple-combine* benchmark which consists of concatenating each datasets and zeroing each missing values.

Validation

Performing a cross-validation directly on the augmented datasets $\mathcal{D}'_{i,\text{G}}$ and $\mathcal{D}'_{i,\text{S}}$ leads to side effects due to the *target-leakage* behavior of both methods. RadialGAN and RadialStyle both encode and decode the target Y in the latent space along with the explicative variables X, in order to preserve the predictive information between X and Y.
Encoding and decoding the target Y introduce a *leakage* of Y in X, which leads to a systematical overfitting when training a predictive model on the augmented datasets $\mathcal{D}'_{i,\text{G}}$ and $\mathcal{D}'_{i,\text{S}}$.
Hence we train $f'_{i,\text{G}}$ and $f'_{i,\text{S}}$ on the augmented (yet leaked) datasets $\mathcal{D}'_{i,\text{G}}$ and $\mathcal{D}'_{i,\text{S}}$, but

[1] https://www.kaggle.com/c/bnp-paribas-cardif-claims-management/

we compute our results on the holdouts $\mathcal{D}_1^{\text{ho}}, \ldots, \mathcal{D}_M^{\text{ho}}$ which were not used for training the transfer models.

We show in our experiments that even though $\mathcal{D}'_{i,\text{G}}$ and $\mathcal{D}'_{i,\text{S}}$ are target-leaked, we still observe a significant performance gain on the prediction of the holdout. This leads to a tradeoff between overfitting the augmented dataset and improving the performance on the holdout.

4.6.2 Dataset Description

Census Income

The Census Income KDD dataset contains weighted census data collected by the U.S. Census Bureau between 1994 and 1995. The prediction task consists of predicting whether or not an individual earns more than $50k per year based on census attributes such as the age, the workclass or the education level, among others. The Census Income dataset contains 199,523 rows and 41 attributes, each attribute can contain missing values and the dataset contains both continuous and discrete attributes.

Facebook Comments

The Facebook Comments dataset contains informations about posts published on a Facebook page, such as the length of the post, the page popularity in which the post was published, the weekdays, to cite a few.
The dataset contains 40,949 posts and 54 attributes. The preciction task consists of predicting the number of comments the post will recieve. Hence, this is a regression task, where the target variable to predict has a significant mass in zero (55%), and is heavy-tailed, with posts having more than one thousand comments.

Claims Management

The Claims Management dataset contains data related to policies and policy holder form the insurer BNP Paribas Cardif. The dataset was proposed as the train set for a Kaggle challenge in 2015. The associated prediction task is to predict a claim category based on features available early on the process.
The dataset contains 114,321 rows and 131 anonymized attributes which can be either continuous or discrete. Some discrete attributes can have a large number of modalities

(some have hundreds of modalities, and one attribute have more than 18,000 modalities).

4.6.3 Results

We compare (in terms of performance gain) each augmentation method along several configurations, e.g. by varying the number n_i and p_i of rows and columns randomly drawn for each split and by changing the dataset $\mathcal{D}$ (Census Income, Bank Marketing, Claims Management).
We repeated each configuration over 20 distinct seeds, in order to measure the mean (and standard deviation) of the performance gain of $f'_{i,\mathrm{G}}$ and $f'_{i,\mathrm{S}}$ over f_i and f_i^c.

Mean Performance Gain: for a random seed $s \in \{1, \ldots, n_s\}$ and a splitted dataset $\mathcal{D}_1^s, \ldots, \mathcal{D}_M^s$, we define the Performance Gain (PG) of a prediction model $g_{i,s}$ trained on $\mathcal{D}'_{i,s}$ over a prediction model trained on $\mathcal{D}_{i,s}$ for a metric $\mathcal{M}$ by the following formula:

$$PG(f_{i,s}, g_{i,s}, \mathcal{M}) := \mathcal{M}(g_{i,s}(X), Y) - \mathcal{M}(f_{i,s}(X), Y), \tag{4.6.1}$$

and we define the Mean Performance Gain by the following formula:

$$MPG(f_i, g_i, \mathcal{M}) := \frac{1}{n_s} \sum_s PG(f_{i,s}, g_{i,s}, \mathcal{M}). \tag{4.6.2}$$

Central Experiment

We start from a *central experiment*, from which we give variants to study the behavior of both RadialGAN and RadialStyle. We use the Census KDD dataset for this experiment, which we split in $M = 3$ sub-datasets, of $n_i = 1000$ rows and $p_i = 10$ attributes $\forall i$.

Figure 4.4 shows the $MPG(f_i, f'_{i,G}, AUC)$ (blue bar), the $MPG(f_i, f'_{i,S}, AUC)$ (yellow bar) and $MPG(f_i, f_i^c, AUC)$ (green bar) for each dataset.
This central experiment illustrates that both RadialGAN and RadialStyle i) have a positive MPG, which means that it gives better performances to train on augmented dataset than to train on a each single splits, ii) give better performances than the Simple-Combine benchmark (except for the RadialGAN on the third dataset), and iii) the RadialStyle outperforms the RadialGAN.

Variants of the Central Experiment

We start from the central experiment previously described, and change some pa-

rameters of the setup of this experiment, to show the behavior of the RadialGAN and RadialStyle. Figure 4.5 illustrates the MPG on the central experiment, except that we drew $p_1 = 10$, $p_2 = 20$ and $p_3 = 30$ attributes instead of $p_1 = p_2 = p_3 = 10$. In this experiment, we see that i) the RadialStyle outperforms the RadialGAN, ii) the MPG of $\mathcal{D}_1$ is higher than the MPG of $\mathcal{D}_2$ and $\mathcal{D}_3$, which is explained by the fact that $\mathcal{D}_2$ and $\mathcal{D}_3$ have more columns to train on, and iii) the RadialGAN does not give better performances than the Simple-Combine benchmark.

Figure 4.6 shows the MPG on the central scenario, except that we splitted $\mathcal{D}$ into $M = 5$ sub-datasets instead of $M = 3$. In this particular experiment, we see that the RadialStyle gives better performances than the Simple-Combine benchmark, and outperforms the RadialGAN, the latter does not give better performances than the Simple-Combine. Note that, by nature, $\mathcal{D}^c$ is less noisy than $\mathcal{D}'_i$, which means that when the number of rows increases linearly, the Simple-Combine's performance also increases.
Also note that in the central experiment we draw $p_i = 10$ attributes $\forall i$, which represents roughly 1/4 of all attributes (recall that the Census KDD dataset contains 41 attributes), meaning that the feature mismatch phenomenon vanishes when M increases and the need of a transfer model is inapropriate, especially if the splits are drawn uniformly (there is no distribution mismatch if the splits are drawn uniformly). The inappropriateness discussed above is depicted in Figure 4.7, where none of RadialGAN and RadialStyle beat the Simple-Combine benchmark.

Now consider the central experiment, except that we vary the number of rows of each dataset. In this experiment, we have $n_1 = 300$, $n_2 = 1000$ and $n_3 = 2000$. Figure 4.8 shows the MPG of both methods for this experiment. We see that our method gives a better performance gain (in terms of AUC gain) than the RadialGAN and the Simple-Combine benchmark (except for the first dataset). Note that the performance gain is inversely proportional to n_i. This is explained by the fact that $\mathcal{D}_1$ has 2,300 rows to train instead of 300, meaning that the additional 2k rows carry a lot of predictive signal. The same applies for $\mathcal{D}_2$ and $\mathcal{D}_3$, but the latter is less sensitive to signal gain due to its relatively large number of rows.
Figure 4.9 shows the MPG of RadialGAN and RadialStyle on the same experiment, except that we now let p_i and n_i varying. In this experiment, we have $p_1 = 10$, $p_2 = 20$ and $p_3 = 30$, and $n_1 = 300$, $n_2 = 1000$ and $n_3 = 2000$. We see that none of RadialStyle and RadialGAN gives a better performance gain than the Simple-Combine benchmark.

Note that the MPG of $\mathcal{D}_1$ is significantly higher than $\mathcal{D}_2$ and $\mathcal{D}_3$ since it has less rows and less columns.

Non-uniform Sampling

For this experiment, we start from the central experiment, except we weight the sampling of $\mathcal{D}_1$ and $\mathcal{D}_2$ along two distinct attributes of $\mathcal{D}$ respectively, and we perform a uniform sampling for $\mathcal{D}$.
This experiment aims at illustrating the ability of both RadialGAN and RadialStyle to handle the distribution mismatch, which is not the case in the central experiment because of the uniform sampling.
Figure 4.10 shows the MPG of this experiment. Note that we did not include the attributes used to weight the sampling of $\mathcal{D}_1$ and $\mathcal{D}_2$ in the random choice of attributes for each seed and for each dataset. We see that RadialStyle outperforms both RadialGAN and the Simple-Combine benchmark.

We also illustrate in Figure 4.11 the same experiment, except that we explicitly give to each dataset the two attributes used to weight the sampling of $\mathcal{D}_1$ and $\mathcal{D}_2$. In this case, the Simple-Combine approach fully observe the two attributes, and therefore is better since it is less noisy by nature. However, RadialStyle still leverages the distribution mismatch and gives better performances on the third dataset.

Non-binary case

All the experiments described above deal with the case where $\mathcal{Y}$ is binary. We now consider the same setup as the central experiment on the Facebook Comments dataset. In this case, the loss $\mathcal{L}$ used in Equation 4.5.3 is the L_2 loss, and the metric used to compute the MPG is the *Root Mean Squared Error* (lower the better):

$$RMSE(y, \widehat{y}) = \sqrt{\frac{1}{N}\sum_i (y_i - \widehat{y}_i)^2}, \tag{4.6.3}$$

where y and $\widehat{y}$ are the ground truth and the predicted vector respectively.

Figure 4.12 shows the MPG on this experiment. We observe that the RadialStyle outperforms both RadialGAN and Simple-Combine, and we also see that the Simple-Combine approach gives performance degradation on $\mathcal{D}_1$ and $\mathcal{D}_2$

Claims Management

All experiments previously described deal with categorical data with a small number of modalities in their attributes. In this experiment, we challenge the RadialGAN and RadialStyle on their ability to handle a categorical attribute with a large number of modalities.
The categorical data is one-hot encoded in each experiment, which means that a categorical attribute with a large number of modalities will result in an input dataset with a large number of columns (in our case, more columns than rows).

For this experiment, we use the same setup as the central experiment on the Claims Management dataset, and force a categorical attribute to be selected on the random choice of attributes. The forced attribute contains 18,210 unique modalities.

Figure 4.13 shows the MPG for this experiment, and Figure 4.14 shows the MPG for the same experiment, except that we do not include the attribute with a large number of modalities. We observe that although none of RadialGAN and RadialStyle achieve better performances than the Simple-Combine, both models leverage predictive information of such an attribute.

Correlations of the predictions

We showed in the previous experiments that the RadialGAN and RadialStyle have distinct behaviors according to the experimental setup. We are now interested in the case where both models have an equivalent quality, i.e. when the MPG of both method is equivalent (e.g. in the central experiment), so that we can enlighten the points where the two methods do not agree.
In other words, if both method have equivalent performances but the predictions of f'_i are widely different whether the augmentation comes from RadialGAN ($f'_{i,G}$) or RadialStyle ($f'_{i,S}$), we believe that a combination of both methods can give a better performance gain than both methods marginally.

Figure 4.15 shows the scatterplots of the predictions of $f'_{i,G}$ (x-axis) and $f'_{i,S}$ (y-axis) for each dataset (one column per dataset) and for 3 distinct seeds of the experiment (one row by seed). We see that the predictions are highly correlated. This is explained by the fact that both $f'_{i,G}$ and $f'_{i,S}$ utilize $\mathcal{D}_i$ when training, which means that both prediction models have a common sub-dataset to train on which do not come neither from RadialGAN nor RadialStyle.

Figure 4.16 shows the scatterplots of the predictions of $f'_{i,G}$ (x-axis) and $f'_{i,S}$ (y-axis), except that we substracted the predictions of f_i to the predictions of $f'_{i,G}$ and $f'_{i,S}$, in order to vanish the correlation which comes from f_i. We see that for each dataset, the predictions are highly correlated, which means that there is no interest in combining the predictions of $f'_{i,G}$ and $f'_{i,S}$.
On the central experiment, the average correlation coefficient between the predictions of $f'_{i,G}$ and $f'_{i,S}$ is .78 for $\mathcal{D}_1$, .83 for $\mathcal{D}_2$ and .80 for $\mathcal{D}_3$.

Computational efficiency

For each experiment presented above, we recorded the total computational time for each method (RadialGAN and RadialStyle) to perform the experiment across the 20 bootstraps. Table 4.1 regroup the total run time of each method according to the setup of each experiment. We see that i) our proposed method is significantly faster than the RadialGAN for each experiment, and ii) that for both methods, the number of splits M and the number of rows n_i for each dataset have the most impact on the computational time.
The total computational time to run the central experiment on 20 distinct seeds depending on the number M of sub-datasets is depicted in Figure 4.17. We see that both method have a quadratical increasing computational time according to M, but the RadialGAN's computational time is significantly higher than RadialStyle.

4.6.4 Discussion

We showed in our experiments that the RadialStyle globally gives a better performance improvement than the RadialGAN. However, recall that we fine tuned each model only once for all seeds.
This means that we did not measure the overall ability of each model to beat the Simple-Combine benchmark, but the cost-efficiency of each method. Indeed, we fixed the cost of tuning of each model, then we run it for 20 distinct seeds, which means that the results previously described are pessimistic.

Thus, it does not mean that the RadialStyle is a uniformly better approach than the RadialGAN, but that it is more stable once the model is tuned, even though RadialGAN could give more performance than both RadialStyle and Simple-Combine.

For each experiment, the column-rate for the random column sampling is a very important parameter to take into account. Indeed, the more the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_M$

Table 4.1 Computational efficiency of the RadialStyle and RadialGAN algorithms according to the experiment setup. The time presented below refers to the total time for each method to perform the experiments on the 20 different seeds. We see that i) our proposed method is significantly faster than the RadialGAN for each experiment, and ii) except for the number of splits M and the number of rows n_i, the experimental setup has no significant influence on the computational time for both methods.
Footnote captions - PF: Preferential sampling, SCNI: Sampling columns not included, SCI: Sampling columns included, LC: Large categorical.

Dataset	M	n_i	p_i	RadialStyle	RadialGAN
Census KDD (central exp)	3	1k each	10 each	**23min**	1h 8min
Census KDD	5	1k each	10 each	**2h 4min**	3h 10min
Census KDD	10	1k each	10 each	**8h 59min**	14h 49min
Census KDD	15	1k each	10 each	**19h 20min**	1d 8h 48min
Census KDD	3	1k each	10, 20, 30	**36min**	1h 15min
Census KDD pf[2], scni[3]	3	1k each	10 each	**36min**	1h 14min
Census KDD pf, sci[4]	3	1k each	10 each	**35min**	1h 13min
Census KDD	3	300, 1000, 2000	10 each	23min	**21min**
Census KDD	3	300, 1000, 2000	10, 20, 30	23min	23min
Facebook comments	3	1k each	10 each	**36min**	1h 12min
Claims management lc[5]	3	1k each	10 each	**35min**	1h 10min
Claims management no lc	3	1k each	10 each	**39min**	1h 11min

will share common attributes, the more Simple-Combine is relevant over a transfer method such as RadialGAN or RadialStyle.
On the other hand the more the datasets' distributions are different, the more relevant it is to use such a transfer method.

We also showed that in some experiments, none of RadialStyle and RadialGAN could give a better performance gain than the Simple-Combine method. This is particularly true when the proportion of common columns between $\mathcal{D}_i$ and $\mathcal{D}_j$ increases. This enlightens the worthiness of performing a domain transfer between each dataset, since the RadialGAN is difficult to train and implement, and does not give systematically better results.

4.7 Conclusion

We proposed a method to perform a multi-domain translation which addresses the feature and distribution mismatch issues while taking into account the characteristics

of the target domain. Our approach is similar to the RadialGAN, i.e. we map all the datasets into a latent space, but we train an additional style transfer model directly into the latent space to better translate from one domain to the target domain.

We proposed an appropriate loss for the style transfer model, which is designed to work with the problem of domain translating with an additional constraint on the fact that we want to preserve the predictive signal lying in each dataset.

We showed that our approach outperforms the RadialGAN, achieving the state-of-the-art performances in domain translation applied to tabular data. Moreover, our approach, besides being more performant, is much faster than the RadialGAN algorithm.

However, in some experiments we showed that our method cannot beat the Simple-Combine benchmark, which motivates our future works. Our approach can be improved by considering an appropriate processing of categorical data, and also by performing a distinct processing according to the nature of data (e.g. text data, high-dimensional datasets etc.). Another possible future extension of our work is to study the worthiness of performing a complex transfer method which does not systematically give a performance improvement.

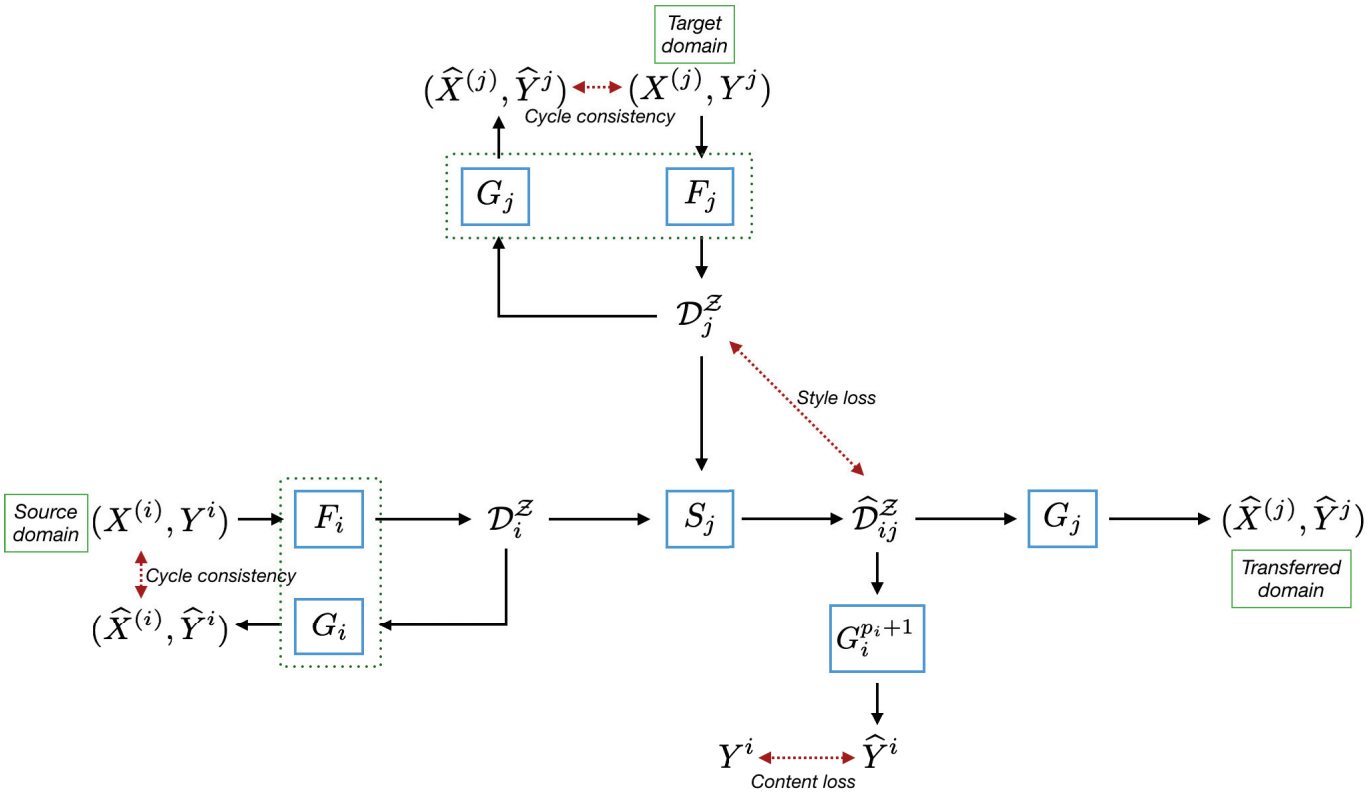

Fig. 4.2 Block diagram of the RadialStyle for transferring the source domain $\mathcal{X}^{(i)} \times \mathcal{Y}$ to the target domain $\mathcal{X}^{(j)} \times \mathcal{Y}$. First, the source and target domains $\mathcal{D}_i = \mathcal{X}^{(i)} \times \mathcal{Y}$ and $\mathcal{D}_j = \mathcal{X}^{(j)} \times \mathcal{Y}$ are mapped to $\mathcal{Z}$, denoted $\mathcal{D}_i^{\mathcal{Z}}$ and $\mathcal{D}_j^{\mathcal{Z}}$ respectively. Then $\mathcal{D}_i^{\mathcal{Z}}$ is translated by S_j to a latent domain $\widehat{\mathcal{D}}_j^{\mathcal{Z}} \approx \mathcal{D}_j^{\mathcal{Z}}$. Finally, $\widehat{\mathcal{D}}_j^{\mathcal{Z}}$ is decoded by G_j to give an approximation of the target domain $\mathcal{D}_j$, denoted $\widehat{\mathcal{D}}_{ij}$.
During the training phase, we first train the encoders and decoders F_i, G_i with the cycle-consistency loss, then we train S_i with the content and style loss.

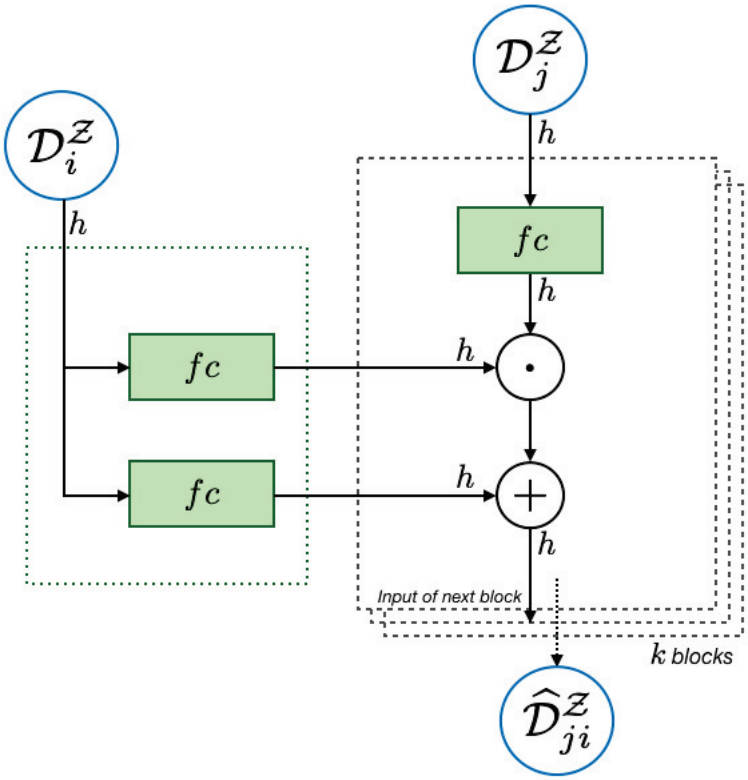

Fig. 4.3 Architecture of the Style Transfer model. Green blocks represent fully connected layers. We input the target domain $\mathcal{D}_i^{\mathcal{Z}}$ to a FiLM generator which predicts the scale and shift parameters α and β, and inject those parameters to a FiLM block. The FiLM block passes the domain $\mathcal{D}_j^{\mathcal{Z}}$ to be transferred through a fully connected layer, then applies the affine transformation. We compose k FiLM blocks, the last output being the transferred latent dataset $\widehat{\mathcal{D}}_{ji}^{\mathcal{Z}}$.

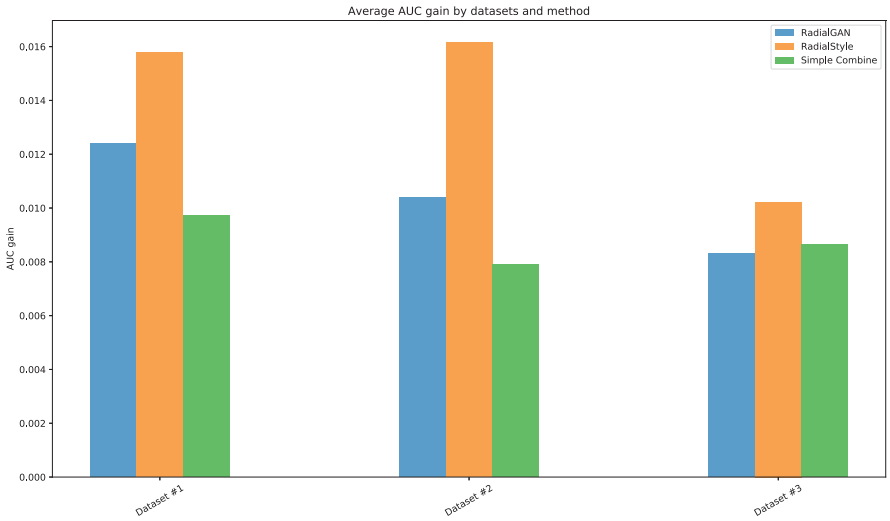

Fig. 4.4 Central experiment: we uniformly sample without replacement the Census KDD dataset into $M = 3$ sub-datasets, each of 10 attributes and 1,000 rows. We see that each method gives better performances than the Simple-Combine benchmark, and the RadialStyle is uniformly better than RadialGAN.

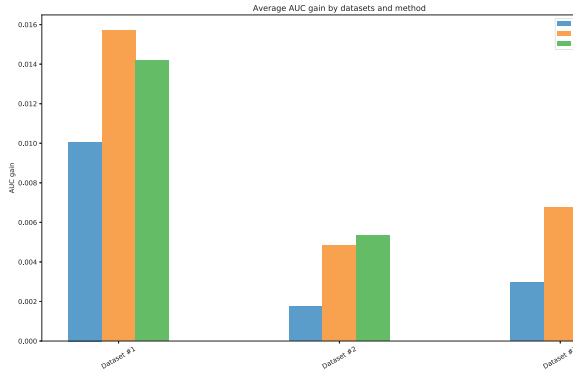

Fig. 4.5 Cental experiment with a varying number of attributes. The datasets have respectively 10, 20 and 30 attributes. We see that i) RadialStyle gives better performances than RadialGAN, and ii) the performance gain on the first dataset is higher because is has less columns.

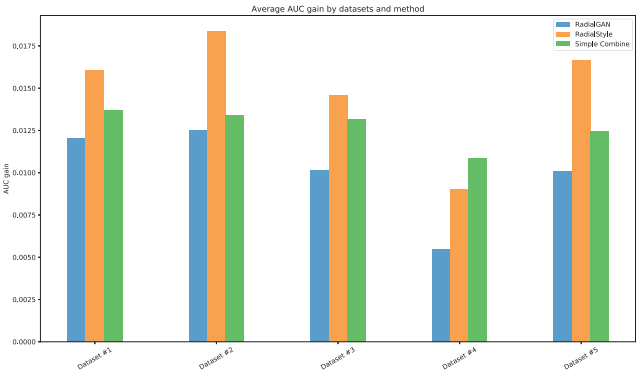

Fig. 4.6 Central experiment with a uniform sampling of $M = 5$ sub-datasets. We see that RadialStyle still outperforms both Simple-Combine benchmark and RadialGAN, but the Simple-Combine benchmark gives better performances when $M = 5$ than when $M = 3$, due to the fact that it is less noisy by nature.

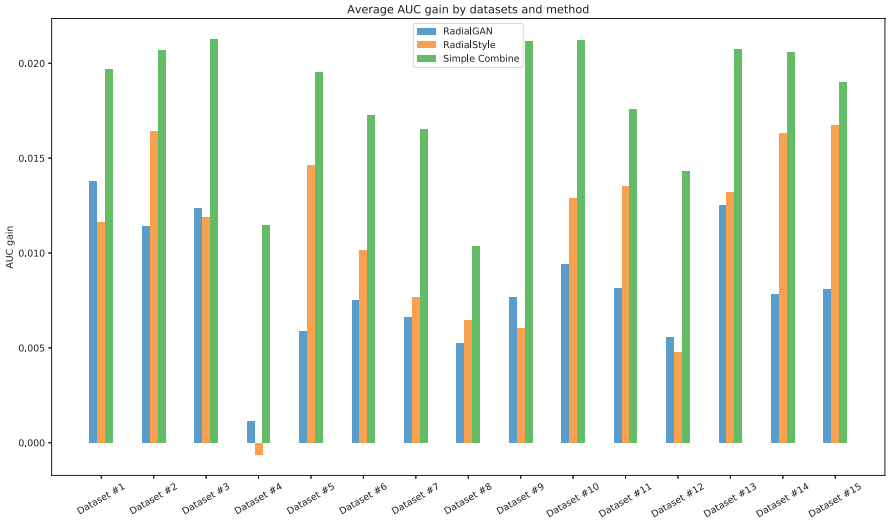

Fig. 4.7 Central experiment with a uniform sampling of $M = 15$ sub-datasets. We see that none of RadialStyle and RadialGAN is able to beat the Simple-Combine benchmark.

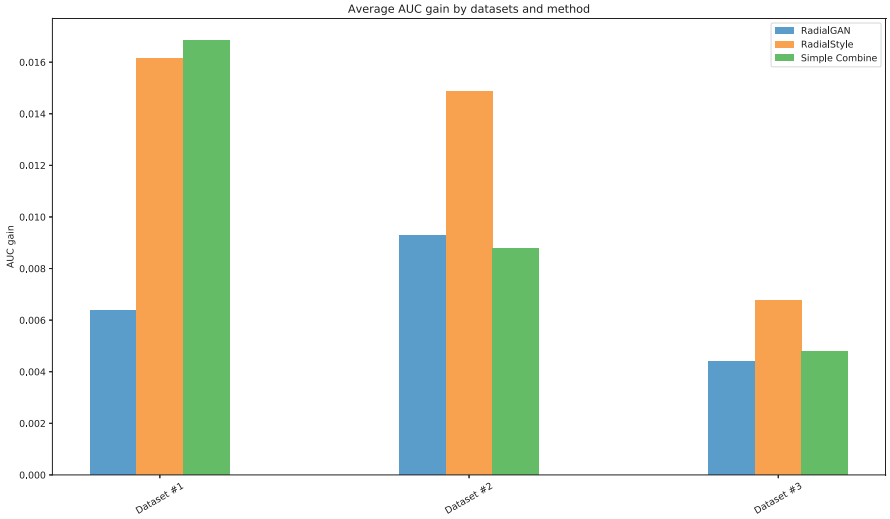

Fig. 4.8 Central experiment with a varying number of rows. The datasets have respectively 300, 1k and 2k rows. We see that the RadialStyle ourperforms the RadialGAN as well as the Simple-Combine benchmark (except for the first dataset). Moreover, we see that for the Simple-Combine and the RadialStyle methods, the performance gain is inversely proportional to the number of rows of each dataset.

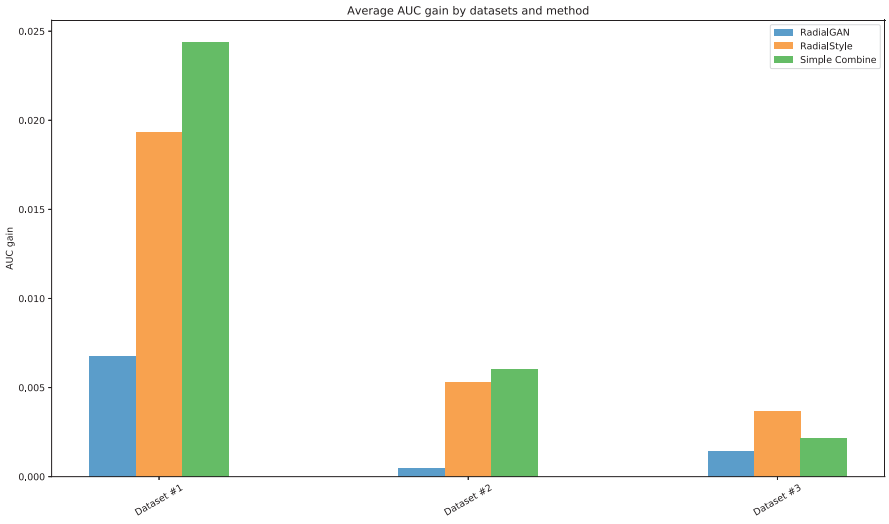

Fig. 4.9 Central experiment with a varying number of rows and columns. The datasets have respectively 300, 1k and 2k rows and 10, 20 and 30 columns. We see that none of RadialStyle and RadialGAN is able to give better performances than the Simple-Combine benchmark (except on the third dataset for RadialStyle). Note that the performance gain of $\mathcal{D}_1$ is significanly higher than $\mathcal{D}_2$ and $\mathcal{D}_3$ since it has less columns and less rows.

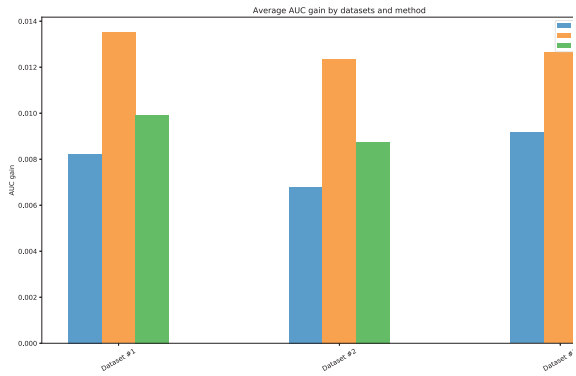

Fig. 4.10 Central experiment with a sampling of $\mathcal{D}$ weighted by two disctinct attributes for $\mathcal{D}_1$ and $\mathcal{D}_2$, and a uniform sampling for $\mathcal{D}_3$. Note that the attributes used to weight the sampling of the first two datasets carry a non-neglictible predictive power. We see that RadialStyle is able to levereage the distribution mismatch induced by the weighted sampling, and that it outperforms both RadialGAN and the Simple-Combine benchmark. Note that we did not include the two attributes used to weight the sampling in the random choice of attributes.

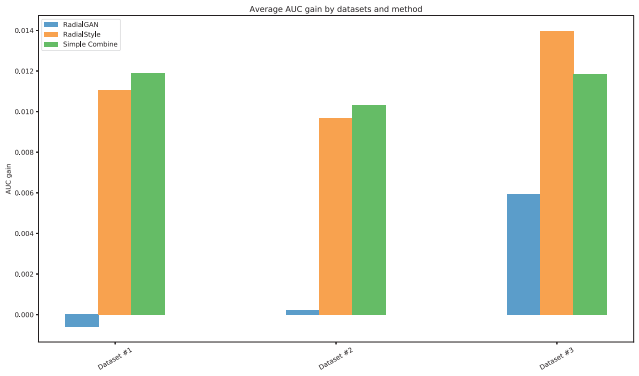

Fig. 4.11 Central experiment with a sampling of $\mathcal{D}$ weighted by two disctinct attributes for $\mathcal{D}_1$ and $\mathcal{D}_2$, and a uniform sampling for $\mathcal{D}_3$. Note that the attributes used to weight the sampling of the first two datasets carry a non-neglictible predictive power. We included for this experiment the two attributes used to weight the first two datasets. We see that RadialStyle is able to levereage the distribution mismatch induced by the weighted sampling, but none of RadialGAN and RadialStyle is able to beat the Simple-Combine benchmark due to the fact that the latter is less noisy.

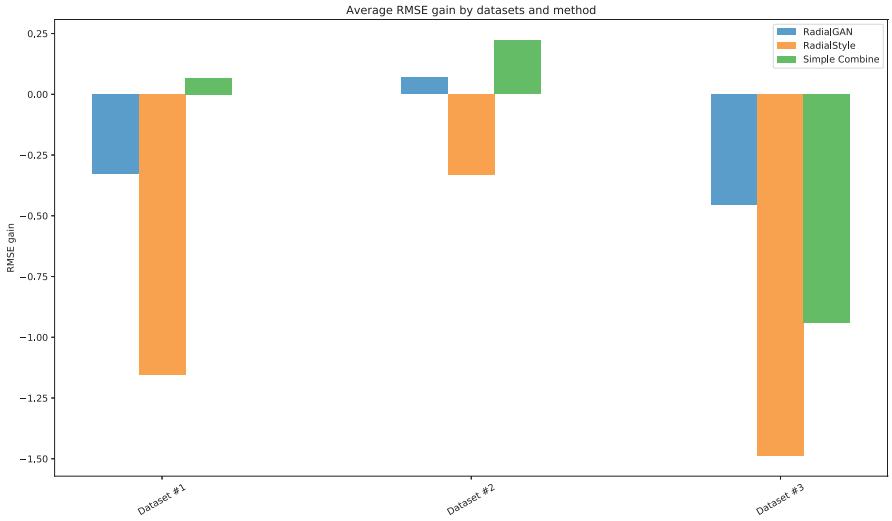

Fig. 4.12 Central experiment on the Facebook Comments dataset. The target to predict is continuous, and the metric used to measure the performance gain is the *root mean squared error* (RMSE), hence the lower the better. We see that RadialStyle gives better performances than the RadialGAn and the Siple-Combine benchmarks.

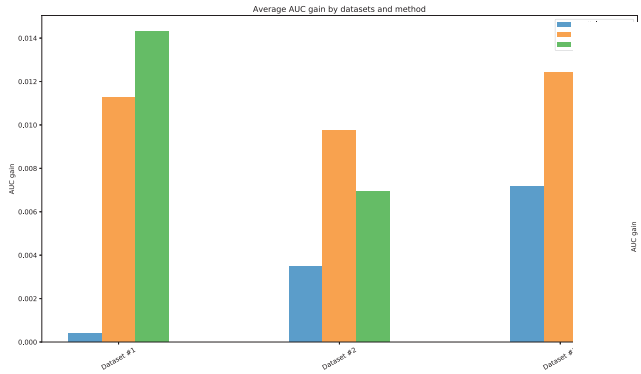

Fig. 4.13 Central experiment on tl Claims Management dataset. In this experiment, we included a categorical attribute which contains 18,200 modalities (this attribute carry a lot of predictive power). We see that the RadialStyle gives better performances than the RadialGAN, but none of these methods is able to uniformly beat the Simple-Combine benchmark. However, the performance gain is still higher than if we do not include the attribute (see Figure 4.14), which means that the methods still leverage predictive signal although too noisy to beat the Simple-Combine benchmark.

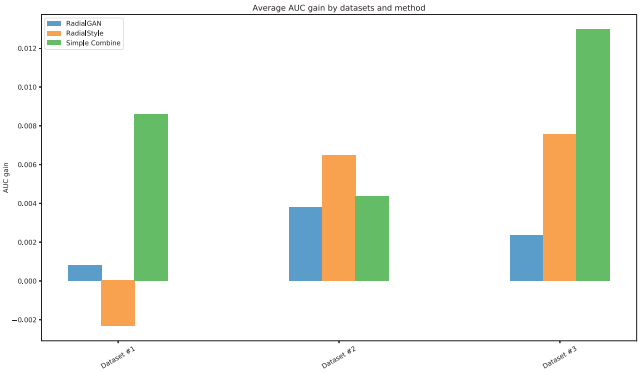

Fig. 4.14 Central experiment on the Claims Management dataset. We did not include the attribute with a large number of modalities in this experiment. We see that none of RadialStyle and RadialGAN is able to beat the Simple-Combine benchmark. Note that this dataset contains less predictive signal than the Facebook Comments and Census KDD datasets. The dataset transfer introduces noise, thus the Simple-Benchmark is hard to beat on this example.

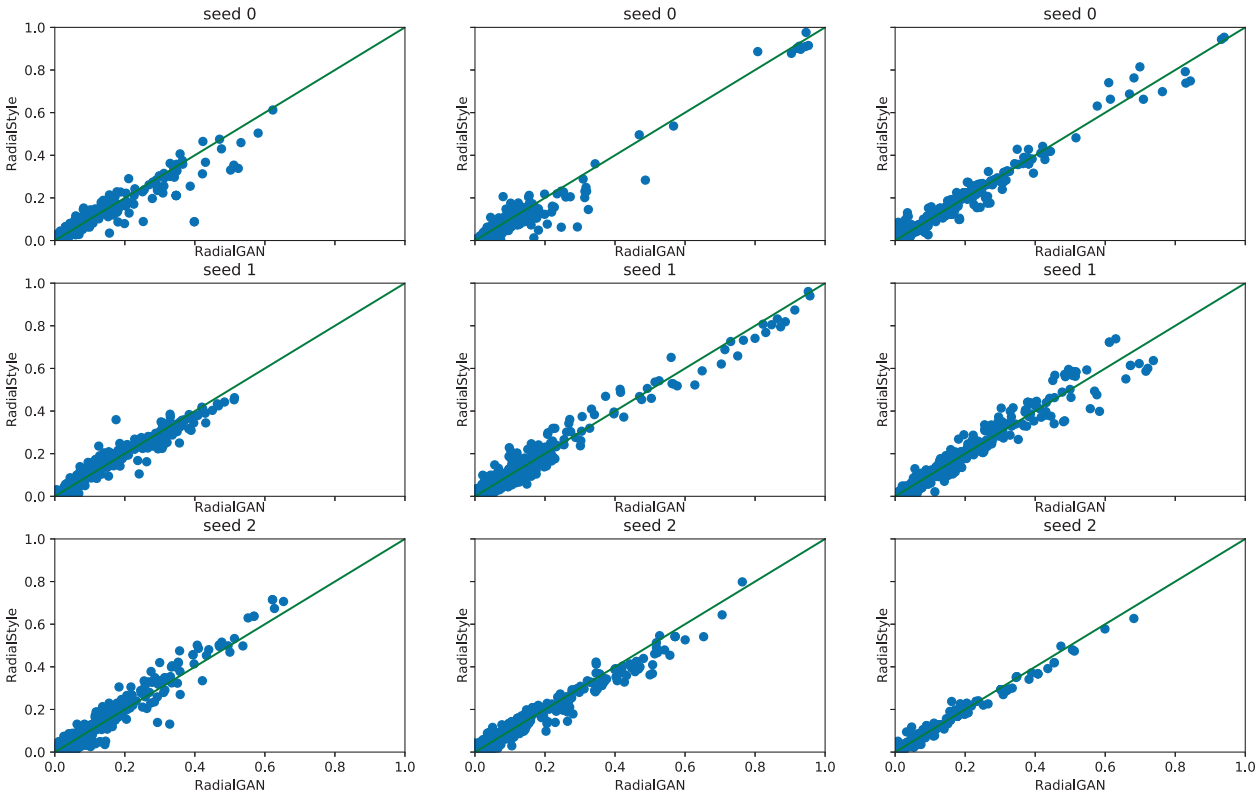

Fig. 4.15 Scatterplots of the predictions of $f'_{i,G}$ (x-axis) and $f'_{i,S}$ (y-axis), for each dataset (one dataset per column), and for 3 distinct seeds of the central experiment (one row per seed). We see that the predictions are highly correlated, which is explained by the fact that both models utilize the raw dataset $\mathcal{D}_i$ (which carry most of the predictive information) as a common part of their training.

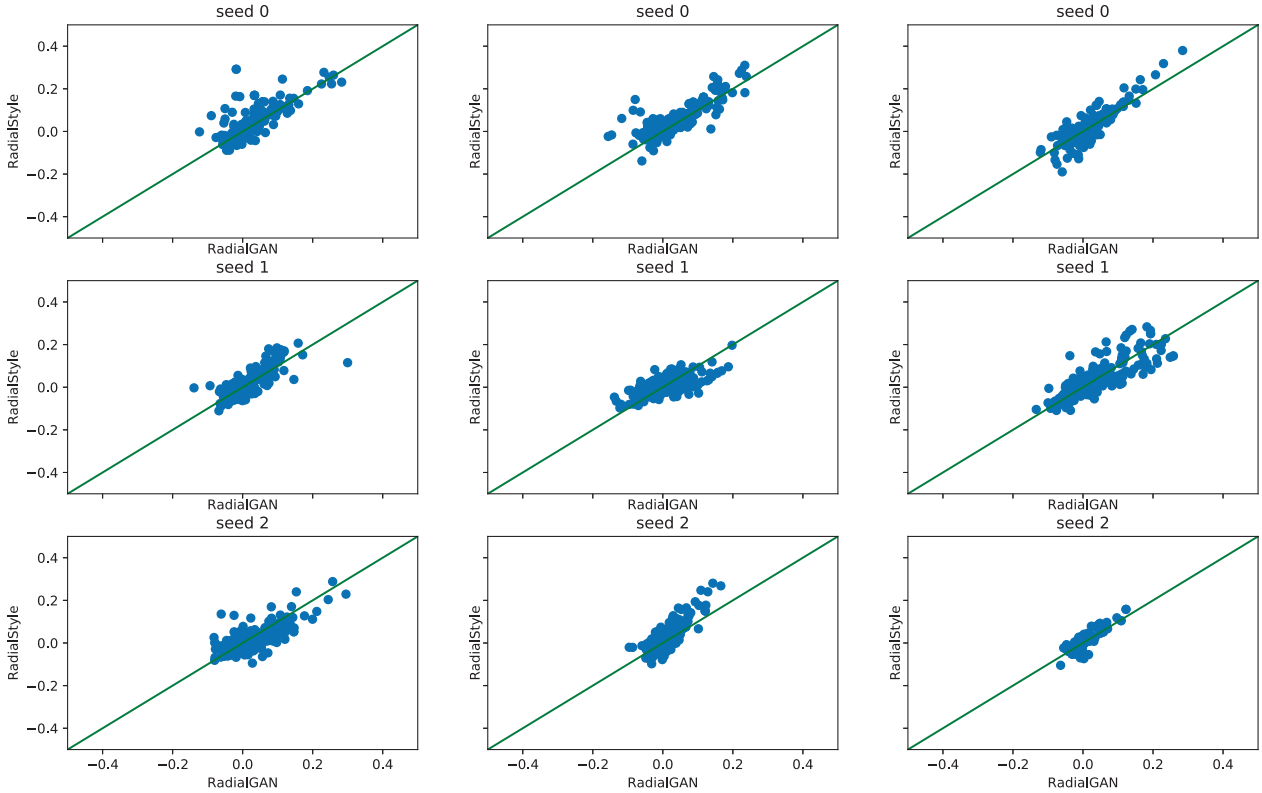

Fig. 4.16 Scatterplots of the predictions of $f'_{i,G}$ (x-axis) and $f'_{i,S}$ (y-axis), for each dataset (one dataset per column), and for 3 distinct seeds of the central experiment (one row per seed). We substracted to each predictions the predictions of f_i, in order to vanish the correlation due to the training of $f'_{i,G}$ and $f'_{i,S}$ on a common dataset $\mathcal{D}_i$ which does not come from RadialGAN nor RadialStyle. We see that the predictions are relatively correlated, but are still quite different, which enlightens a potential interest of combining both approaches.

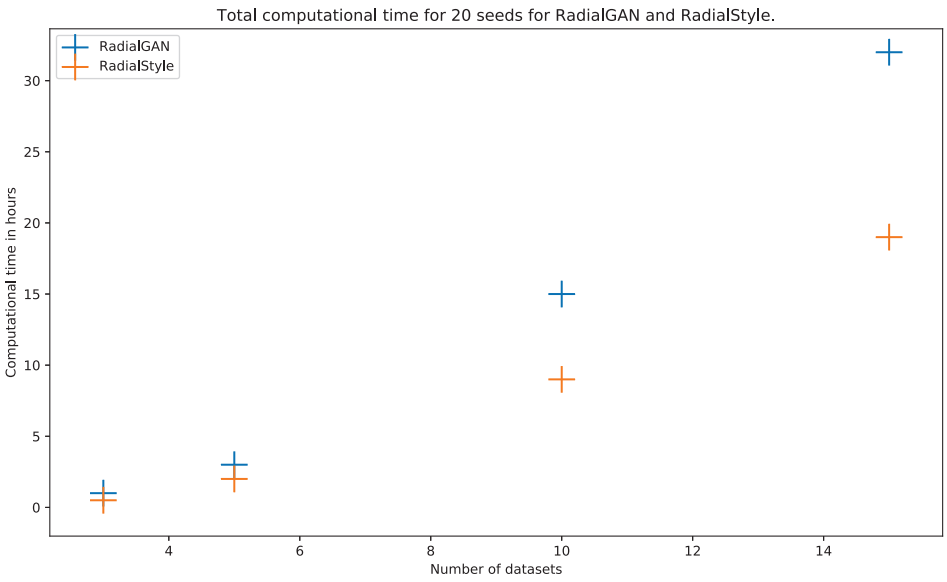

Fig. 4.17 Computational time of RadialGAN (blue marks) and RadialStyle (yellow marks) to run the central experiment on 20 distinct seeds. We vary the number M of sub-datasets in the x-axis and record the computational time in hours on the y-axis. We see that both method have a quadratical increasing computational time, but the RadialGAN is significantly longer to train than the RadialStyle.

Chapter 5

Autoregressive Implicit Quantile Networks for Time Series Generation

5.1 Introduction

Time series analysis gathered significant interest in a wide range of applications such as weather forecasting, stock market analysis, sales forecasting, among many others. We distinguish two problem-dependent focuses on time series analysis : *forecasting* and *generating*. Forecasting refers to the estimation of conditional expectations whereas generating focuses on the estimation of conditional distributions.
The interest of forecasting (predicting) a time series is to estimate what is most likely to realize in the future values (path) of the time series. For example, forecasting answers the question of *"what the weather* will *be like tomorrow (or any other future date)?"*. On the other hand, generating a time series refers to the fact that we need to estimate the distribution of the observed stochastic process, allowing us to consider a large amount of potential future realizations of the time series. For example, it gives multiple appropriate answers to the question of *"what the weather* could *be like tomorrow?"*, including extreme scenarios.

Time series generation is usually done by setting a parametric structure and generating observations along that model, allowing us to get as many possible future paths as needed. Structural models have an old history in econometrics literature, e.g. linear time series [Box and Jenkins, 1970, Brockwell and Davis, 1991, Hamilton, 1994]. Such

models are widely used in practice, e.g. by the insurance industry to build Economic Scenarios Generators (ESG), in order to evaluate the "Best Estimate liabilities". It can also be used to simulate different weather scenarios coupled with electricity consumption to optimize the energy distribution. It is useful to generate GPS data for traffic prediction to prevent traffic jams. It also opens the way for video generation since it takes into account the temporal dependencies between each observation, to cite a few. Moreover, being able to learn the distribution of any time series may overcome legal issues by working with generated samples instead of personal or sensitive data.

Deep generative models such as Generative Adversarial Networks (GAN) [Goodfellow et al., 2014], Variational Auto-Encoders (VAE) [Kingma and Welling, 2013], and autoregressive density estimation (PixelCNN) [Oord et al., 2016] have recieved a lot of attention from the deep learning community, leading to significant recent advances in many application domains, such as image generation [Radford et al., 2015, Karras et al., 2017], text generation [Guo et al., 2017, Yu et al., 2017] and raw audio generation [van den Oord et al., 2016a]. However, the deep learning literature dealing with time series is still scarce, and the few significant works mainly focus on time series prediction (i.e. forecasting).
Recent work in deep learning for time series forecasting seeks to model complex non-linear patterns. In particular, the use of Recurrent Neural Networks (RNN) [Hochreiter and Schmidhuber, 1997] and Convolutional Neural Networks (CNN) [LeCun et al., 1998] has led to outstanding performances in time series prediction [Giles et al., 2001, Krishnan et al., 2017, Borovykh et al., 2017], achieving the state-of-the-art performances in multivariate, asynchronous time series [Binkowski et al., 2017, Che et al., 2018].

Deep generative modeling applied to time series have not revieved enough attention from the deep learning community. Recent work includes [van den Oord et al., 2016a] who introduced the WaveNet architecture to perform raw audio generation, [Esteban et al., 2017] who applied GANs to generate real-valued medical time series and [Zhang et al., 2018] who generate smart grid data by using GANs. All of these works successfully managed to estimate the underlying distribution of the stochastic process, hence generating realistic new potential future paths for the time series.
To the best of our knowledge, no other thorough work has been proposed to estimate the family of conditional distributions of a time series. This is explained by the lack of a reliable data-agnostic criterion to assess the quality of a generative model applied

on time series. The usual metrics to assess the generation quality are the Inception Score (IS) [Salimans et al., 2016] and Fréchet Inception Distance [Heusel et al., 2017], which both rely on an Inception network [Szegedy et al., 2016] pretrained on ImageNet [Deng et al., 2009], which is not usable with time series.

In this paper, we propose an autoregressive implicit quantile network (AIQN) approach to estimate the conditional distributions of a time series. Our method is inpired by the WaveNet model [van den Oord et al., 2016a] except that i) it estimates the quantile function instead of the likelihood and ii) it is designed to work with continuous data. The latter point is of high importance in time series generation since it allows us to accurately estimate the distribution tails of the stochastic process, which cannot be done with the WaveNet model.
Our approach has strong implications for performing time series simulation without any structural model, opening the way to estimate complex patterns in the time series while being able to generate new samples along its distribution.

We first challenge the ability of our approach to learn and simulate simple time series, e.g. AR, ARMA, VAR, ARCH/GARCH models (see [Hamilton, 1994]) and give an illustration of our method on a simulated high dimensional time series.
We also evaluate our approach on two real-world datasets: a subsample of a financial portfolio of equities, and GPS car data trajectories.

We propose to evaluate the generation quality by focusing on three key time series characteristics: i) the Auto-Correlation Function (ACF), which quantifies the temporal dependencies in the time series, ii) the Quantile-Quantile plot between real and generated marginals, which measures the ability of the model to fit the true distribution, and iii) the MMD statistic [Gretton et al., 2008] which measures the closeness of two multivariate, high dimensional distributions.
We also compute the Cross Auto-Correlation Function (CACF) between each dimensions of the time series in the multivariate case.

We show that in almost every example, our approach provides excellent results with respect to these characteristics, and is able to learn complex properties and time dependencies of the time series, while taking into account the limited size of the training sample. We also address the problem of high dimensional time series by proposing a way to embed a stochastic process to a latent space.

In this paper, we make the following contributions:

- we also propose an AIQN architecture with residual blocks designed for 1D data, coupled with an additional gating mechanism and an extra residual connection,
- we perform an importance sampling on the target quantiles and add multivariate inputs to be more accurate on the estimation of the marginal distribution and on the time dependencies,
- we show that our approach outperforms the state-of-the-art models,
- we propose a framework to embed a time series in a latent space of low dimension to adress the problem of high dimension.

The outline of this paper is organized as follows. Section 5.2 gives the context of this paper and on related works on time series and deep generative modelling, Section 5.3 gives details about structural time series, the metrics we use to assess the performance of the generative models and on the loss we use for our approach. Section 5.4 describes our proposed approach and Section 5.5 propose an adaptation of our model on high dimensional time series. Section 5.6 illustrates our approach and compare it with the state-of-the-art model on time series generation, and Section 5.7 discuss the results.

5.2 Related work

Literature on time series is rich and has led to a broad range of works in econometrics which makes extensive use of stochastic structural models such as AR, ARMA, VAR and ARCH/GARCH, among others. Such structural models can be used to forecast a time series as well as to sample new observations, which is widely used in practice to perform Monte Carlo experiments.
Deep learning literature on time series is scarce, and mainly focuses on the forecasting exercise. [Binkowski et al., 2017] proposed the use of CNN to modelize non-linear patterns of the offset and volatility of time series, and achieved state-of-the-art performances in financial time series forecasting. [Che et al., 2018] proposed a deep hierarchical hidden Markov model to capture autoregressive patterns and led to excellent results in time series forecasting.

Literature on deep generative modelling is quite recent but is rich and has led to

several advances in the way of modelling a target distribution. Variational Auto-Encoders (VAE) [Kingma and Welling, 2013] approximate the density by maximizing a tractable lower bound of the likelihood. Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] directly learn an implicit density by setting a two-player zero-sum game between two networks: a generator G and a discriminator D which aims at distinguishing real and fake (generated) samples while G aims at generating data realistic enough to induce D in error. The whole stake of GAN is to find a Nash equilibrium between G and D. On the other hand, PixelCNN architectures [Oord et al., 2016, van den Oord et al., 2016b] provide autoregressive models which learn tractable densities by performing masked convolutions with residual gated blocks, and allow to estimate the conditional distribution of each observed point given its past.

Literature on deep learning for time series generation is even scarcer, and mainly focuses on the use of GANs to directly modelize the distribution. [Esteban et al., 2017] applied GAN using LSTM [Hochreiter and Schmidhuber, 1997] network for the generator and discriminator to generate real-valued medical sequences, and introduced the use of the Maximum Mean Discrepancy (MMD) criterion [Gretton et al., 2008] to evaluate the performance of the GAN. On the other hand, the WaveNet model [van den Oord et al., 2016a] can be used to generate time series since raw audio data is basically a time series. However, the latter does not fit when the dimension of the time series is greater than the number of realizations (i.e. the high dimensional case), and is desingned to work with discrete data, which is not the case for a real-valued time series. Note that it is of significant interest to give a good estimation of the tails of distribution for a time series, and the WaveNet does not take such a behavior in its architecture.

The use of the Kullback-Leibler (KL) divergence as a part of the loss function has been recently challenged [Dabney et al., 2018a, Dabney et al., 2018b], leading to the use of a quantile loss to introduce a new kind of networks: Implicit Quantile Networks (IQN). The adaptation of the conditional PixelCNN architecture [van den Oord et al., 2016b] to IQN network for image generation has been proposed by [Ostrovski et al., 2018] to extend the IQN to autoregressive models, leading to a new family of generative models: Autoregressive Implicit Quantile Networks (AIQNs). AIQN architectures, relying on the reparameterization of a quantile τ drawn from a uniform distribution, allow to directly estimate the quantile function of a distribution without the need to analyze the likelihood by optimizing a KL divergence. Thus, in this work, we leverage the

AIQN architecture to generate time series, and propose to use additional criteria to evaluate our model.

5.3 Background

Let $(X_t)_{t\in\mathbb{Z}}$ be a multivariate stochastic process in $\mathbb{R}^d$. Forecasting the future values of $(X_t)_{t\in\mathbb{Z}}$ is equivalent to estimating the conditional expectation of its future values

$$\mathbb{E}[X_{t+h}|X_t, X_{t-1}, \ldots], \ \forall h > 0, \tag{5.3.1}$$

whereas generating new samples is equivalent to estimating its conditional multivariate distributions by considering the product of its univariate conditional distributions

$$p(X_{t+h}, \ldots, X_{t+1}|X_t, X_{t-1}, \ldots) = \prod_{j=1}^{h} p(X_{t+j}|X_{t+j-1}, \ldots),$$

and generate samples along the estimated distributions. This is equivalent to estimating the conditional quantile functions of $(X_t)_{t\in\mathbb{Z}}$.

5.3.1 Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) [Gretton et al., 2008] is a powerful non parametric tool to compute a distance between two multivariate, high dimensional distributions p, q from which we observe two finite samples $x := (x_1, \ldots, x_n), y := (y_1, \ldots, y_m)$. For a class of functions $\mathcal{F}$, the MMD is defined by the following formula:

$$MMD(p, q) := \sup_{f\in\mathcal{F}} \left(\mathbb{E}_x[f(x)] - \mathbb{E}_y[f(y)]\right). \tag{5.3.2}$$

An estimator of the MMD can be computed by replacing the expectations by their empirical estimators along x and y:

$$\widehat{MMD}(p, q) := \sup_{f\in\mathcal{F}} \left(\frac{1}{n}\sum_i f(x_i) - \frac{1}{m}\sum_i f(y_i)\right). \tag{5.3.3}$$

Note that due to the fact that we replaced the expectations by their natural estimator directly in the *sup*, the MMD estimator in Equation 5.3.3 has an upward bias. However, [Gretton et al., 2008] showed that the MMD can be written in terms of kernel functions if $\mathcal{F}$ is taken as the unit ball of a universal Reproducting Kernel Hilbert Space (RKHS). More formally, the MMD^2 can be written using kernel functions:

$$MMD^2(x,y) = \mathbb{E}_{x,x'}[k(x,x')] - 2\mathbb{E}_{x,y}[k(x,y)] + \mathbb{E}_{y,y'}[k(y,y')], \tag{5.3.4}$$

where x' (resp. y') is an independant copy of x (resp. y) following the same distribution as x (resp. y), and k is a kernel, usually Gaussian. From that fromula, an unbiased estimator can be build by replacing the conditional expectations from Equation 5.3.4 by their natural estimator:

$$\widehat{MMD}^2(x,y) = \frac{1}{n(n-1)}\sum_{i\neq j} k(x_i,x_j) - \frac{2}{mn}\sum_{i,j} k(x_i,y_j) + \frac{1}{m(m-1)}\sum_{i\neq j} k(y_i,y_j).$$

More details about the construction of the unbiased MMD estimator and the link between Equations 5.3.2 and 5.3.4 can be found in [Gretton et al., 2008].

MMD non parametric test

The quality of the generated time series can be assessed by a non parametric test using the MMD^2 statistic. The null hypothesis H_0 of such a test is the equidistribution of p and q, i.e. $H_0 : p = q$. We estimate the empirical distribution of the MMD^2 statistic under H_0 by computing bootstraped replicas of $MMD^2(\tilde{x},\tilde{y})$, where $\tilde{x},\tilde{y} \sim r$ and r is a mixture of p and q (e.g. $r = \frac{1}{2}p + \frac{1}{2}q$) and then compute the empirical p-value of $MMD^2(x,y)$ under H_0.
Note that we also can take $r = p$ or $r = q$ equivalently, but a mixture of p and q allows us to bootstrap $\tilde{x}$ and $\tilde{y}$ with more observations.

Figure 5.1 illustrates the sensitivity of the test $T : (H_0; MMD^2)$ when p is a standard bivariate normal distribution of mean $\mu = (0,0)$ and variance matrix $\Sigma = I_2$ (identity), and q is a standard bivariate normal distribution of mean $\mu = (\mu_1,\mu_2)$ and identity variance matrix. We see that when μ_1,μ_2 slightly increases (from 0 to 0.2), the test rejects H_0.

In practice, this test is very sensitive, and tends to systematicallty reject H_0 since the generated samples do not match exactly the target distribution. For that matter, we rather compute the average MMD^2 value over many replicas of the same experiment then computing the p-value of the non parametric test.

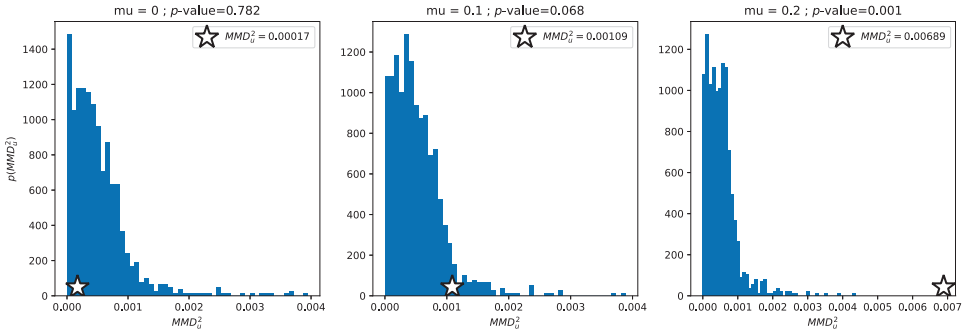

Fig. 5.1 Illustration of the non-parametric test $T : (H_0; MMD^2)$. Recall that H_0 corresponds to the equidistribution of p and q, and that the distribution of MMD^2 under the null hypothesis is computed by drawing iid replicas of $MMD^2(\tilde{x}, \tilde{y})$, where $\tilde{x}, \tilde{y} \sim r$, and r is a mixture of p and q. In this example, p and q are both standard bivariate normal distributions, except that we slightly change the mean of q from $(0, 0)$ to $(.2, .2)$. The star value corresponds to the MMD^2 value between the two samples $x \sim p$ and $y \sim q$ to test, i.e. $MMD^2(x, y)$. We see that the test is very sensitive, detecting slight changes in the distribtuions.

5.3.2 Auto-Correlation Function

Besides evaluating our model with the MMD^2 statistic, we propose to consider an additional criterion to decompose its ability to estimate the time series' dependencies in and between each dimensions of $(X_t)_{t\in\mathbb{Z}}$, which is not possible with the MMD^2. For a stationary stochastic process $(X_t)_{t\in\mathbb{Z}}$ of mean μ and variance σ^2, the Auto-Correlation Function (ACF) for a lag integer $h \geq 0$ measures the correlation between X_t and X_{t-h} $\forall t \in \mathbb{Z}$, i.e.

$$ACF(h) := \frac{\mathbb{E}[(X_t - \mu)(X_{t-h} - \mu)]}{\sigma^2}.$$

When $(X_t)_{t\in\mathbb{Z}}$ is multivariate, the Cross Auto-Correlation Function (Cross-ACF) for dimensions i and j of $(X_t)_{t\in\mathbb{Z}}$ is defined as

$$CACF(h)_{ij} := \frac{\mathbb{E}[(X_t^{(i)} - \mu_i)(X_{t-h}^{(j)} - \mu_j)]}{\sigma_i \sigma_j},$$

where μ_i (resp. μ_j) is the mean of the i-th (resp. j-th) component $(X_t^{(i)})_{t\in\mathbb{Z}}$ (resp. $(X_t^{(j)})_{t\in\mathbb{Z}}$) of $(X_t)_{t\in\mathbb{Z}}$.

In our experiments, we assess the quality of the generated sample by computing

the ACF and the CACF of both $(X_t)_{t\in\mathbb{Z}}$ and the generated sample for multiple values of h (usually h vary from 0 to 50).

5.3.3 Structural Time Series Models

An Auto-Regressive Moving Average (ARMA) model of order p and q is defined by the following recursion:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t,$$

where $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a white noise. Such a linear model is widely used in practice, and we challenge the ability of our model to estimate this specific structure without any prior. The Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) model of order p and q focuses on the conditional variance σ_t^2 of a stochastic process $(X_t)_{t\in\mathbb{Z}}$ over time, and is defined as

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i X_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2,$$

where $X_t = \sigma_t \varepsilon_t$, and $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a white noise. Although we input the raw stochastic process $(X_t)_{t\in\mathbb{Z}}$ to the model, we expect it to modelize the time dependencies of the squared stochastic process $(X_t^2)_{t\in\mathbb{Z}}$.
The Dynamic Conditional Correlation (DCC-) GARCH model is a multivariate version of the GARCH model, and can be defined as:

$$X_t = H_t^{1/2} \varepsilon_t,$$

where $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a vector of white noises, $H_t = D_t R_t D_t$, D_t is the diagonal matrix of conditional standard deviations of X_t at time t, and R_t is the conditional correlation matrix for X_t at time t. As for the GARCH model, we show that our model is able to learn the ACF and CACF of the squared data.

5.3.4 Auto-Regressive Quantile Networks

Quantile regression is a method used to estimate a specific point in a distribution. It is performed by optimizing the quantile loss function $\rho_\tau(u) := u(\tau - \mathbb{1}_{\{u\leq 0\}})$ [Koenker and Hallock, 2001], which takes as arguments a target quantile τ and an estimation error u which is the difference between the observation and the estimated quantile.

Although quantile regression is not new, using it as the loss function of a neural network is recent and opened the way to the use of AIQNs [Dabney et al., 2018b, Dabney et al., 2018a, Ostrovski et al., 2018] for generative modelling. Such approaches give an extra input $\tau \sim \mathcal{U}([0,1])$ to the network to be reparametrized directly on the quantile function.
Once the model is trained, the generation process takes as input i) a new realization $\tau \sim \mathcal{U}([0,1])$, seen as the expected quantiles to be generated, and ii) a time series, either empty or partially realized, to be generated. If we input a non-empty time series, then we generate a future path conditionally to the given realizations.

In practice, AIQN models actually use an alternative version of the quantile loss, the Huber loss [J. Huber, 1964], which allows the gradient to scale with the magnitude of the error u under a threshold κ:

$$\rho_\tau^\kappa(u) = \begin{cases} \frac{|\tau - \mathbb{1}_{u \leq 0}|}{2\kappa} u^2, & \text{if } |u| \leq \kappa \\ |\tau - \mathbb{1}_{u \leq 0}|(|u| - \frac{1}{2}\kappa), & \text{otherwise.} \end{cases} \tag{5.3.5}$$

In this paper, we propose an AIQN architecture designed for time series, which directly learns the underlying conditional quantile functions of the stochastic process $(X_t)_{t \in \mathbb{Z}}$.

5.4 Model

The model presented in this work follows the architecture of a regular gated conditional PixelCNN [van den Oord et al., 2016b], except that:

- we substitute the residual gated block by a 1D version and we add an additional residual connection at the end of the network in order to better learn the distribution shape,
- our additional input τ follows a beta distribution instead of a uniform distribution, for reasons we detail later in this section.

The benefit of using an AIQN for time series is that once the model is trained, it allows us to generate paths along any wanted range of quantiles τ, which means that it is possible to generate a large range of distinct scenarios (like extreme ones), which is of high interest in the financial industry.

5.4.1 Estimating the auto-correlations

This steps consits of estimating the auto-correlations and cross auto-correlations of $(X_t)_{t\in\mathbb{Z}}$. To do so, we use the AIQN architecture (PixelIQN) proposed in [Ostrovski et al., 2018] as our baseline for our 1D adaptation of the residual block, and train the model to optimize the Huber loss defined in Equation 5.3.5.

Our AIQN model is built upon several residual blocks in which we input the target quantile τ for reparameterization, and pass the input time series through masked convolutions such that each observation is a function of the past observations. Masked convolutions, illustrated in Figure 5.2, consist of zeroing the weights of the kernel which correspond to future observations, allowing i) the model to be causal and ii) the parallelization of the training phase.

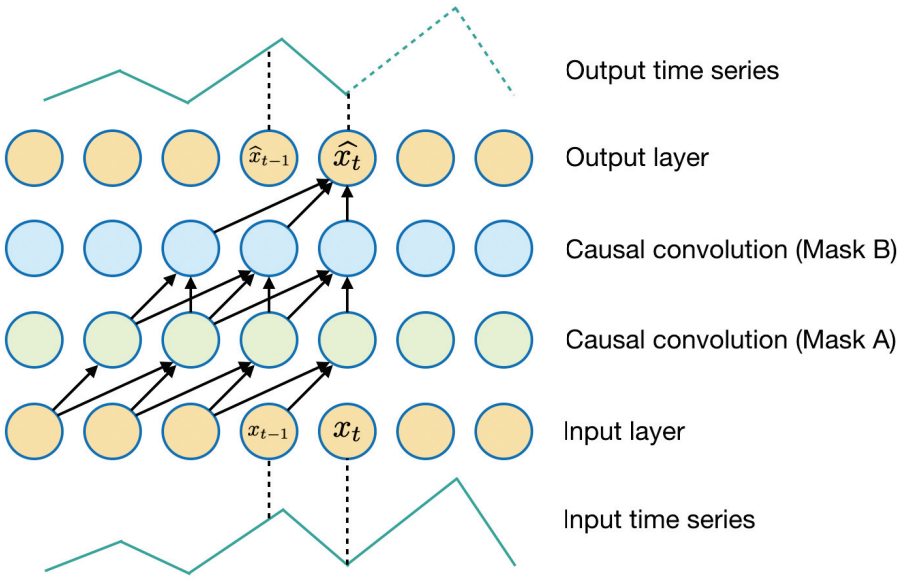

Fig. 5.2 Illustration of 1D masked convolutions. Mask A convolution does not use the current observation in order to ensure that the model is causal. Mask B convolution allows us to use the encoding of the current observation.

The amount of signal passed through the output of the residual block number b is computed along a gating mechanism of the following form:

$$\widehat{x_t} = tanh(W_{b,f} * x + U_{b,f} * \tau) \odot \sigma(W_{b,g} * x + U_{b,g} * \tau),$$

where $W_{b,f}$ and $W_{b,g}$ are $k \times 1$ kernels for an uneven integer k, $U_{b,f}$ and $U_{b,g}$ are 1×1 kernels, $x = (x_{t-\frac{k-1}{2}}, \ldots, x_t, \ldots, x_{t+\frac{k-1}{2}})$, σ is the sigmoid function, $*$ is the convolution operator and $\odot$ is the Hadamard (element-wise) product. Figure 5.3 illustrates the 1D AIQN.

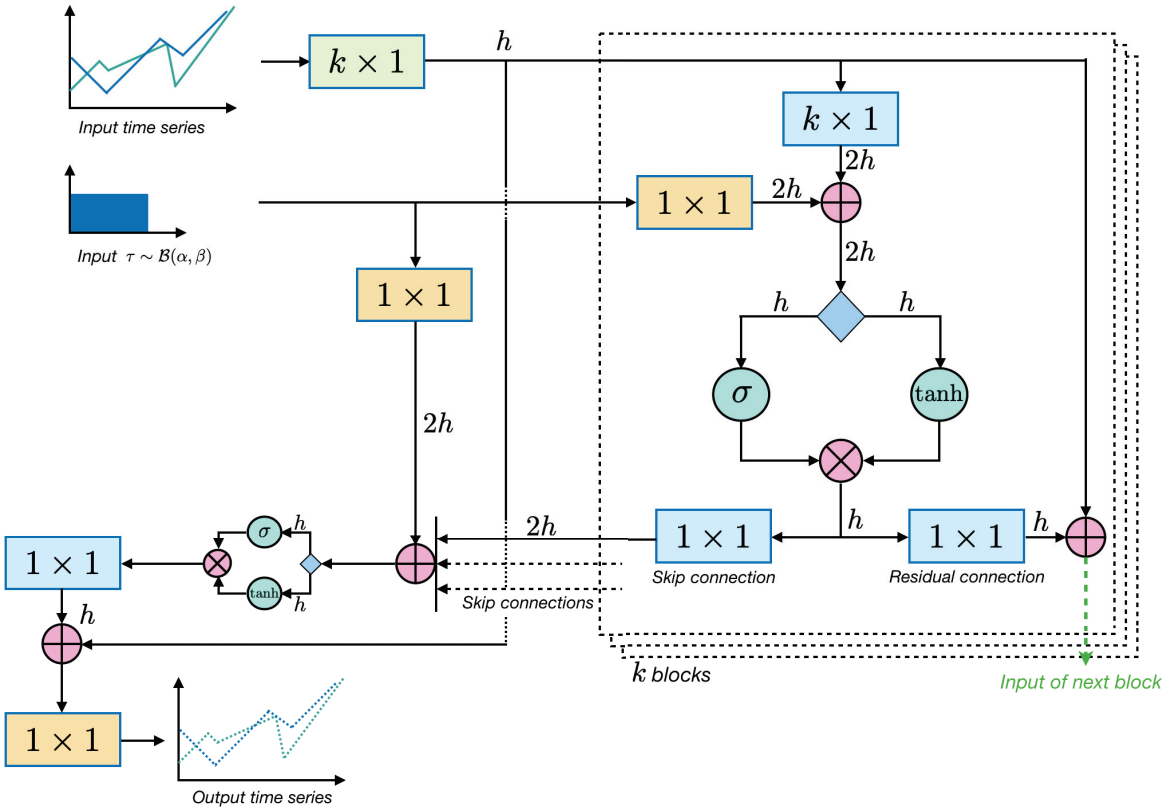

Fig. 5.3 Illustration of the AIQN architecture we use. Green blocks are mask A convolutions, blue blocks are mask B convolutions, yellow blocks are regular convolutions. An additional gating mechanism and a residual connection on the output layer have been added to better estimate the marginal distribution of the time series.

Validation setup

Let $x = (x_1, \ldots, x_n)$ be a realization of the stochastic process $(X_t)_{t \in \mathbb{Z}}$. In practice, since we often have only one realization, we split x into sub-vectors of fixed size $m << n$ by sliding a window of size m through x, in order to build the training, validation and holdout datasets. Such a split introduces dependencies between each input of the model, but on the other hand it overcomes the fact that we have a dataset with only one row.

5.4.2 Estimating a continuous distribution

Generative models tackling on image related tasks usually provide a probability on a range of integer color levels (from 0 to 255) for each pixel.
In our case, the output of the model is a value in a continuous, non-compact, high dimentional space (usually $\mathbb{R}^d$). This implies that generative models on time series have to cover a potentially unbounded support for $(X_t)_{t\in\mathbb{Z}}$, which can raise issues for the model to learn some quantiles in the frontier of the output space.

For that reason, we propose to perform an importance sampling and use the beta distribution $\mathcal{B}(\alpha, \beta)$ for τ instead of a uniform distribution (see Figure 5.4).
For example, when $(X_t)_{t\in\mathbb{Z}}$ is heavy tailed, setting $\alpha = \beta = 1/2$ gives a privileged direction for the model to learn the distribution tails (i.e. quantiles close to 0 and 1). Note that the regular quantile regression can still be performed by fixing $\alpha = \beta = 1$, which corresponds to the uniform distribution.
Besides, such an importance sampling for τ is done only in the training phase. For the generation phase, we input a uniform distribution for τ to the model.

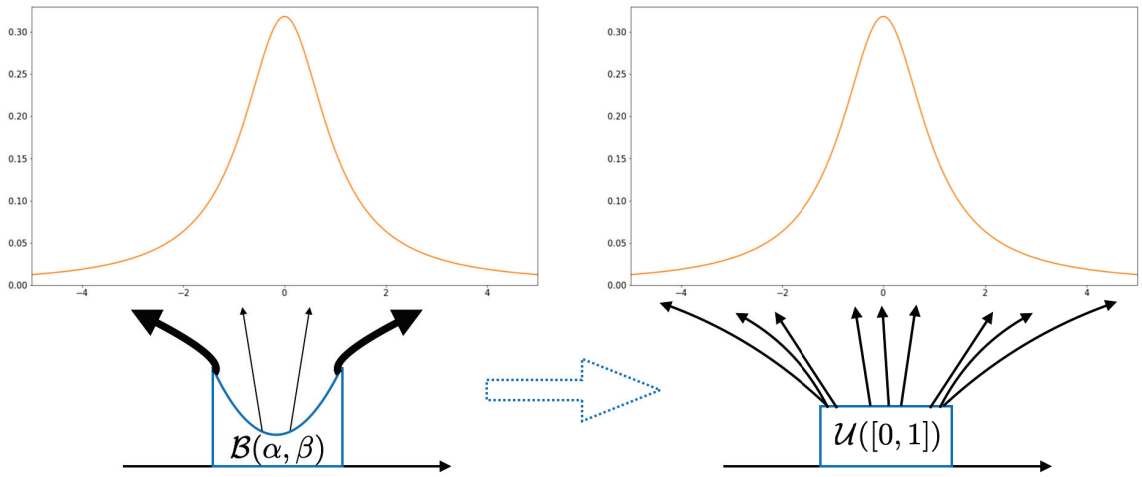

Fig. 5.4 Example of the use of beta distribution for τ when $\alpha = \beta = 1/2$. It gives more weight to the quantiles corresponding to the distribution tails for reparameterization in the learning phase (left). On the sampling phase (right), we still use the uniform distribution.

An important thing to note here is that in this paper, we only have at our disposal the conditional distributions, and by definition we do not observe the marginal distributions. This explains why the model may have difficulties estimating the marginal distributions, but it gives excellent results on the estimation of conditional distributions. We partially

overcome this issue by giving additional channels to the input of the model, which aims at giving more information on the marginals.

5.5 Linear High Dimensional Time Series

In this section, we study the behavior of our approach in high dimension. Let $(X_t)_{t\in\mathbb{Z}}$ be a stochastic process in $\mathbb{R}^d$, and let $x = (x_1, \dots, x_m)$ be m realizations of $(X_t)_{t\in\mathbb{Z}}$. Training an AIQN model (as defined in Section 5.4) directly on x when $m >> d$ is very effective, and we illustrate this in Section 5.6. However, when $m << d$, the AIQN model is no longer able to optimize the loss defined in Equation 5.3.5. This is explained by the fact that the number of dimensions d is greater than the number of available realizations m of $(X_t)_{t\in\mathbb{Z}}$, which means that the model cannot be identified in a reasonable amount of time, hence the need to sum up the information contained in $(X_t)_{t\in\mathbb{Z}}$ in lower dimension.

Such an issue is not adressed in image processing since an image (or equivalently a video) is an object of low dimension (usually 3 dimensions corresponding to the *red*, *green* and *blue* channels). On the other hand, this issue is adressed in the Natural Language Processing (NLP) literature, and the most effective way to lower the dimension is to let a model estimate a latent reprezentation of the input data (namely word embedding in NLP) wiht a minimum loss of information, then to work directly through the latent space.

In order to make the AIQN work in high dimension, we encode $(X_t)_{t\in\mathbb{Z}}$ in a latent space $\mathcal{Z}$ of dimension $k << m$, then we train the AIQN directly through $\mathcal{Z}$ to estimate the auto-correlations, then we rebuild the time series in the original feature space.

Let $F : \mathbb{R}^d \to \mathcal{Z}$ and $G : \mathcal{Z} \to \mathbb{R}^d$ be an *encoder* and a *decoder* respectively, and let $\mathcal{Z}$ be a real-valued space of dimension $k << d$. We jointly train F and G to map $(X_t)_{t\in\mathbb{Z}}$ in $\mathcal{Z}$ and map back to the original input domain $\mathbb{R}^d$ with a minimum loss of information. More specifically, we expect F and G to i) correcly encode the auto-correlations of $(X_t)_{t\in\mathbb{Z}}$, and ii) to correctly rebuild the support of $(X_t)_{t\in\mathbb{Z}}$.We do so by minimizing the L_2 loss between $(X_t)_{t\in\mathbb{Z}}$ and $G(F((X_t)_{t\in\mathbb{Z}}))$:

$$\min_{F,G} \sum_t \mathbb{E}[||X_t - G(F(X_t))||_2^2], \tag{5.5.1}$$

where $||.||_2$ is the L_2 norm. Figure 5.5 depicts the block diagram of the training phase of F and G.

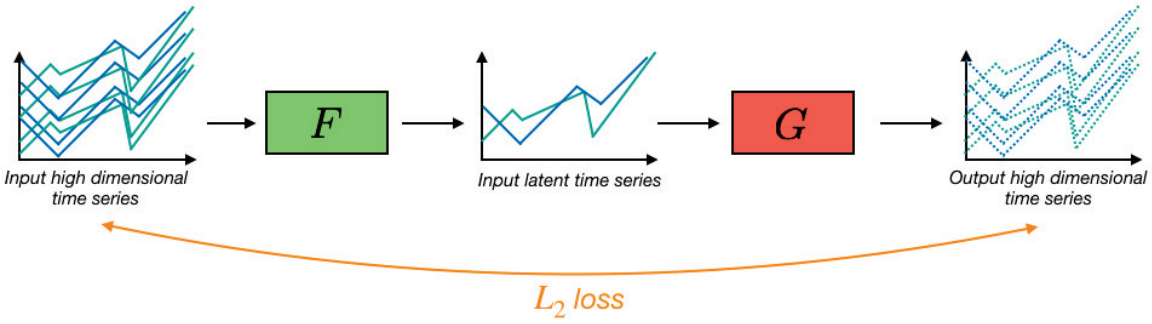

Fig. 5.5 Auto encoder of the time series. F aims at encoding a high dimensional stochastic process $(X_t)_{t\in\mathbb{Z}}$ to a latent real-valued space $\mathcal{Z}$ of lower dimension k without losing the auto-correlation and cross auto-correlation information. G aims at rebuilding the time series form its latent reprezentation.

Once the auto-encoders F and G are trained, we train the AIQN model as defined in Section 5.4 directly through the latent space. The whole training phase is then decomposed in the following two steps:

1. train F and G to respectively sum up $(X_t)_{t\in\mathbb{Z}}$ in a lower dimension space $\mathcal{Z}$ and rebuild $(X_t)_{t\in\mathbb{Z}}$ from $\mathcal{Z}$ to $\mathbb{R}^d$,
2. estimate the auto-correlations directly through $\mathcal{Z}$.

Figure 5.6 shows the diagram of the training phase of the AIQN in high dimension.

The hyperparameter $k = dim(\mathcal{Z})$ can be chosen by cross-validation or by making an additional assumption on the fact that most of the variance of $(X_t)_{t\in\mathbb{Z}}$ can be summarized in k dimensions.

5.6 Experiments

In this Section, we evaluate the sampling quality of our model on generated data from models presented in Section 5.3.3. We also have at our disposal a financial dataset, composed of a subsample of an equities portfolio, and GPS records of car trajectories.

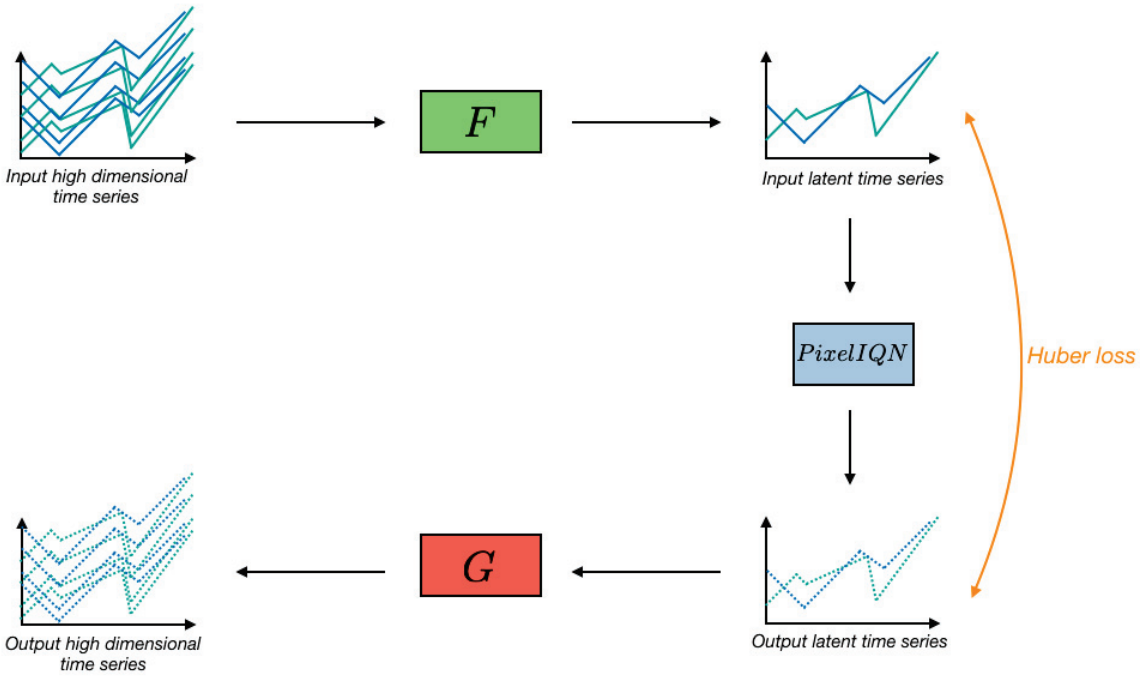

Fig. 5.6 Block diagram showing the training phase of the AIQN in high dimension. We first encode $(X_t)_{t\in\mathbb{Z}}$ in $\mathcal{Z}$ by using F, then we train the AIQN direclty through $\mathcal{Z}$, then we rebuild the time series by using G.

For all the examples presented in this Section, we compare our model with the state-of-the-art generative model for time series, the RGAN introduced in [Esteban et al., 2017]. For the sake of fairness for each comparison, we used exactly the same datasets for training, validation and testing for each model, and we illustrate our results on the test set, which was not used neither for training nor tuning.

We compute the MMD^2 statistic between the real and fake (i.e. generated) samples, drawn from our AIQN model and from the RGAN model. Then, we compare it to the MMD^2 value between two subsamples drawn from the real dataset. This allows us to compare the MMD^2 value of each model against a proxy of the "perfect" model, i.e. the model which generates the data.

As explained in Section 5.3.2, stationary time series may be characterized by the following quantities: i) the autocorrelations and the cross autocorrelations for multivariate time series, ii) the marginal distributions. We expect our model to learn and generate samples which reproduces i) and ii).

For a stochastic process $(X_t)_{t\in\mathbb{Z}}$ in which we observe one realization $x := (x_1, \dots, x_n)$, we denote by $s := (s_1, \dots, s_m)$ and $z := (z_1, \dots, z_m)$ two generated samples of length m drawn respectively from AIQN and RGAN. In all our experiments, we considered $m = n$.
We denote by x^h the lagged values of order h of x, i.e. $(x^h)_i = (x_i, \dots, x_{i-h})$. We compute the MMD^2 on s^h and z^h against x^h. The motivation behind giving the lagged values as input of the MMD^2 is to take into account the time dependencies of x, s and z when computing their distribution distance. We also compute a proxy of the MMD^2 from a "perfect" model by computing $MMD^2_{perfect} = MMD^2(x_{:\tilde{t}}, x_{\tilde{t}:})$, where $\tilde{t} = (t-h)/2$, $x_{:\tilde{t}} = (x_1, \dots, x_{\tilde{t}})$ and $x_{\tilde{t}:} = (x_{\tilde{t}}, \dots, x_n)$.

To compare the ability of AIQN and RGAN to estimate the conditional distributions of the time series, we also compute the MMD^2 between x^h and the realigned samples $s^h_{(r)}$ and $z^h_{(r)}$, defined (with a slight abuse of notation) as $s^h_{(r)} := Q_x(F_s(s^h))$, where Q_x is the empirical quantile function of x, and F_s is the empirical Cumulative Distribution Function (c.d.f) of s.

5.6.1 Simulated data

We evaluated the ability of our model to generate samples drawn from time series presented in Section 5.3.3. More precisely, we considered the AR(1) model (i.e. ARMA(1,0)) with a positive and a negative coefficient, the ARMA(1,1) model, the GARCH(1,1) model and the DCC-GARCH (with dimension $d = 3$) model, all with Gaussian white noise $(\varepsilon_t)_{t\in\mathbb{Z}}$.

Table 5.1 summarizes the mean and standard deviations of the MMD^2 values $MMD^2_{perfect}$, $MMD^2(x^h, z^h)$ and $MMD^2(x^h, s^h)$ for each dataset (synthetic and real data). We computed those values on raw generated samples, as well as on samples on which we realigned quantiles, such that the MMD^2 only measures the time dependencies. It shows that, in almost every example, our architecture outperforms the RGAN benchmark. Moreover, the standard deviations of the MMD^2 values of our model are generally lower than the RGAN, which means that our method provides more stable results.
Figures 5.8, 5.9, 5.10 and 5.11 illustrate the ACF and CACF on some of the examples. It shows that the autocorrelations of our model are more accurate than the RGAN.

Figures 5.10 and 5.11 show the ACF and CACF for the DCC-GARCH example, on squared samples. We can see that our approach gives reasonably good estimations of

Table 5.1 MMD^2 statistics. For each model, we present the average and standard deviation (in brackets) of the MMD^2 statistic (the lower the better) computed along 50 simulations of each experiment (except for real data, for which we cannot have more than one realization). Results with the symbol $(*)$ are computed on realigned quantiles, in order to measure the ability of each model to estimate the autocorrelations. The best results -except the "perfect" model ("Real" column)- are denoted in bold font.

Model	h	Real	AIQN	RGAN
AR(1) coef $\frac{1}{2}$ $(*)$	5	0.000865 (0.001043)	**0.002054 (0.001306)**	0.006024 (0.005145)
AR(1) coef $-\frac{1}{2}$ $(*)$	5	0.000338 (0.000430)	**0.001935 (0.001793)**	0.003200 (0.002924)
ARMA(1,1) $(*)$	5	0.000827 (0.000853)	**0.003143 (0.001068)**	0.004142 (0.002866)
GARCH $(*)$	3	0.000616 (0.001313)	**0.001912 (0.001540)**	0.004129 (0.002628)
DCC-GARCH $(*)$	3	0.000203 (0.000267)	**0.000337** (0.000389)	0.000586 **(0.000337)**
AR(1) coef $\frac{1}{2}$	5	0.000865 (0.001043)	**0.030233 (0.012370)**	0.054767 (0.040990)
AR(1) coef $-\frac{1}{2}$	5	0.000338 (0.000430)	0.034425 **(0.012898)**	**0.032442** (0.038371)
ARMA(1,1)	5	0.000827 (0.000853)	**0.019256 (0.010948)**	0.035712 (0.044200)
GARCH	3	0.000616 (0.001313)	**0.029475 (0.016888)**	0.044454 (0.047202)
DCC-GARCH	3	0.000203 (0.000267)	**0.003337 (0.002567)**	0.075253 (0.005377)
Financial data	1	0.006594 (-)	**0.036849 (-)**	0.072214 (-)
Financial (squared)	1	0.003596 (-)	**0.052112 (-)**	0.095664 (-)
GPS Trajectories	10	0.001954 (-)	**0.002361 (-)**	0.024256 (-)

the significant autocorrelation on the squared observations, whereas RGAN fails to estimate such quantities.

Linear High Dimensional Time Series

We illustrate our approach on two simulated high dimensional time series of dimension $d = 200$. We simulate the i-th component with the following formula:

$$X_t^{(i)} = \beta_i Y_t + \varepsilon_t^{(i)}, \tag{5.6.1}$$

where Y_t is an auto regressive model AR(1) with coefficient 1/2 and variance 4/3 (the variance of an AR(1) model with corefficient φ is $1/(1-\varphi^2)$), $\beta_i \sim \mathcal{U}([0,2])$ is a coefficient which can be interpreted as the amount of observed signal (the higher β_i the higher the observed signal), and $\varepsilon_t^{(i)}$ is a Gaussian white noise of variance 1/15 independant of Y_t. Note that each dimension i has a separate white noise $\varepsilon_t^{(i)}$ and that $\varepsilon_t^{(i)} \perp\!\!\!\perp \varepsilon_t^{(j)}$ for $i \neq j$.

We challenge the ability of our model to i) estimate the auto-correlations of each

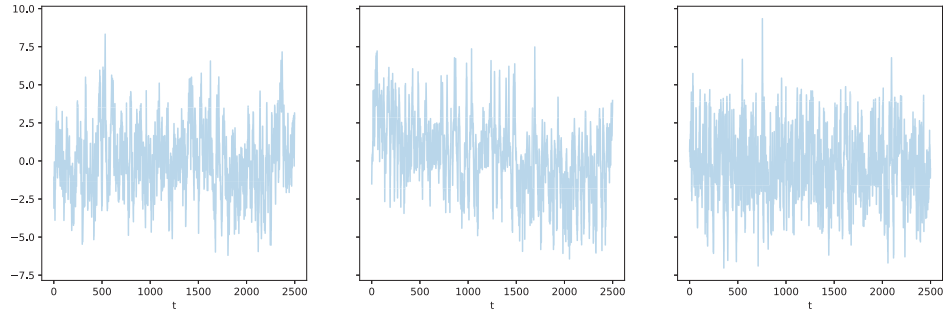

Fig. 5.7 Plots of real (left), AIQN (middle) and RGAN (right) samples. The real sample follows an auto regressive model AR(1) with coefficient .9.

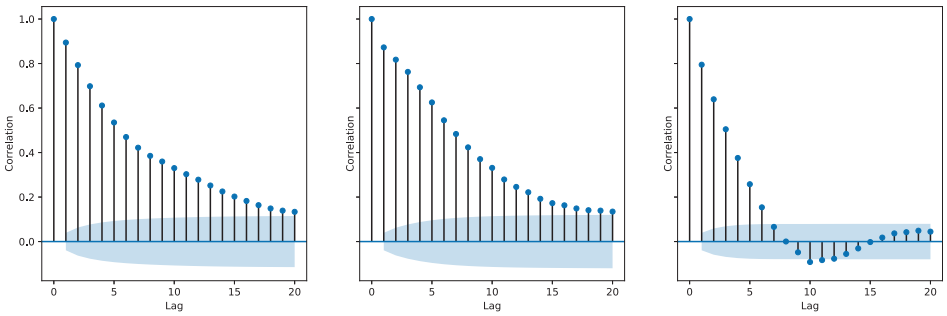

Fig. 5.8 ACF of real (left), AIQN (middle) and RGAN (right) samples when the underlying model is an AR(1) with coefficient .9. We can see that the RGAN's autocorrelations are less accurate than AIQN.

component of $(X_t)_{t\in\mathbb{Z}}$, and ii) accurately estimate the support of each dimension. Note that the approach defined in Section 5.5 cannot accurately rebuild the noise term $\varepsilon_t^{(i)}$. For that matter, we rebuilt such a noise term by performing a bootstrap of the observed error term $e^{(i)} := (X_t)_{t\in\mathbb{Z}} - G(F((X_t)_{t\in\mathbb{Z}}))^{(i)}$, then add the bootsrapted empirical error to $G(F((X_t)_{t\in\mathbb{Z}}))$.

Figure 5.12 shows the ACF of the generated sample (using our method) the ACF of the true sample on the first ten dimensions. On this example, we do not show the ACF of the RGAN since the latter did not converge. We see that our model is able to accurately estimate the ACF function on most of the dimensions. On the other hand, Figure 5.13 shows the associated scatterplot between the real sample and the generated sample on the first ten dimensions. We see that our approach gives excellent estimations of the support of the generated sample.

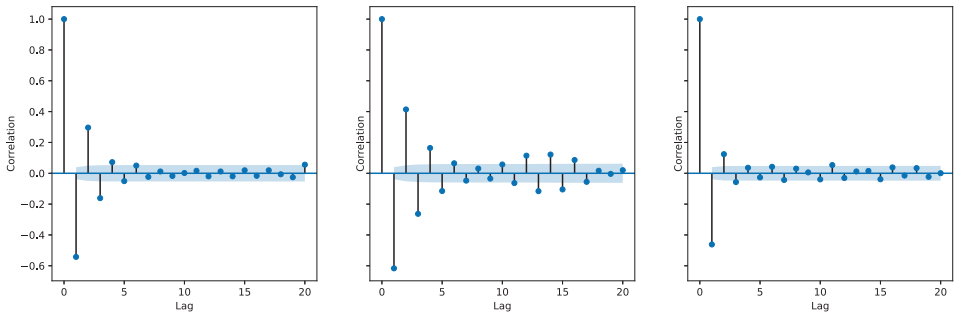

Fig. 5.9 ACF of real (left), AIQN (middle) and GAN (right) samples on the AR(1) coefficient $-1/2$ model.

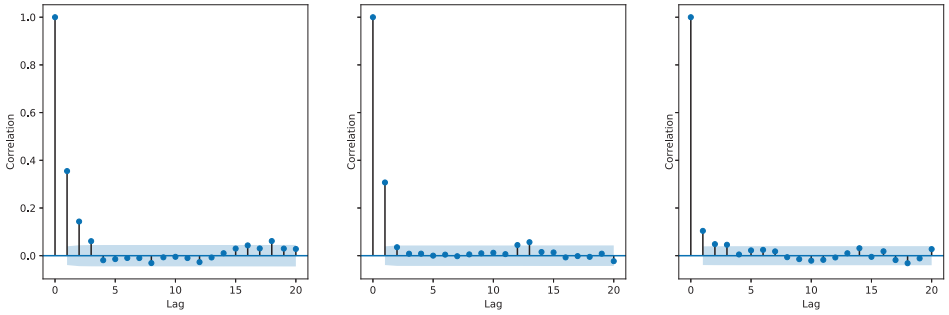

Fig. 5.10 ACF of real (left), AIQN (middle) and GAN (right) squared samples on the DCC-GARCH model.

We also challenge our model to estimate the distribution of a high dimensional stochastic process $(X_t)_{t\in\mathbb{Z}}$ of dimension $d = 200$ when the error term is heteroskedastic. We simulated the i-th dimension of such a phenomenon according to the following formula:

$$X_t^{(i)} = \beta_i Y_t + \varepsilon_t^{(i)}, \tag{5.6.2}$$

where Y is an autoregressive model AR(1) with coefficient 7/10, β_i is a random coefficient uniformly draw between 0 and 2, and $\varepsilon_t^{(i)}$ is a Gaussian white noise which variance depends on the values of $X_t^{(i)}$. In this experiment, the variance of $\varepsilon_t^{(i)}$ is equal to $(X_{t-1}^{(i)}/10)^2$. Note that we performed the same bootstrap as in the previous experiment to empirically rebuild the error term $\varepsilon_t^{(i)}$.

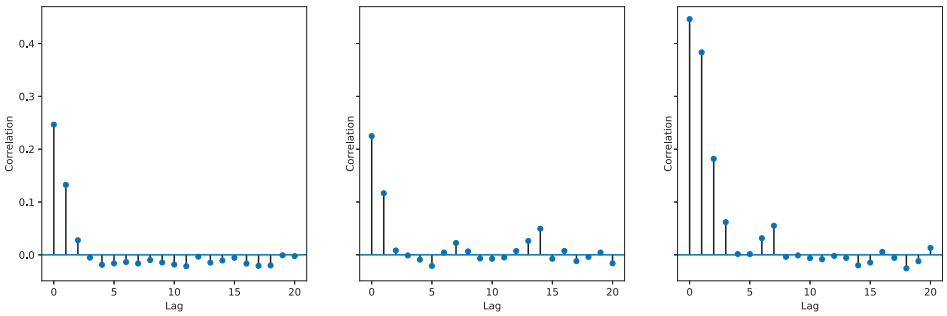

Fig. 5.11 Cross-ACF of real (left), AIQN (middle) and GAN (right) squared samples on the DCC-GARCH model.

Figures 5.14 and 5.15 show respectively the ACF and the scatterplots of the generated sample and the real sample for the first ten dimensions. We see that our approach gives accurate estimations of the auto correlations and is able to correctly estimate the support of $(X_t)_{t\in\mathbb{Z}}$, even if the latter is heteroskedastic.

5.6.2 Financial data

We test our method on a real-world dataset which is a subsample of a financial portfolio[1]. We have at our disposal the daily end-of-day values for each equity, recorded from 2012-01-01 to 2017-12-31. We train our model on increments of the end-of-day prices from 2012-01-01 to 2014-12-31 and generate data from 2015-01-01 to 2017-21-31 and compare the generated sample to the real one.

Figure 5.16 shows the Q-Q plots of the real sample against the generated samples drawn from AIQN (circle blue markers) and RGAN (triangle green markers) respectively. We see that our model gives a better quantile alignment, which is confirmed by the corresponding MMD score of Table 5.1 (on non-squared sample). Moreover, Figure 5.17 shows that the RGAN's squared sample estimates significant autocorrelations whereas there is no such autocorrelations in the real sample. This is also observed on most of the other equities, whereas our model respects that criterion.

[1] This dataset is freely available with the Quandl API for academic purposes only. https://www.quandl.com/.

5.6.3 GPS data

Telematics data are very promising for car insurers to measure the risk of a policy holder as he drives, allowing to analyze its driving habits and adapt the insurance premium according to the risk. Generative modelling on such data implies that we expect the model to learn the driving style of each driver.
For this experiment, we have at our disposal a dataset recording at each second the GPS coordinates of 1,753 individual car drivers. For each driver, we have 200 individual trajectories.
Works on such data have been done to extract statistics which enlight the user's driving style by computing the v-a heatmap [Wüthrich, 2017], which displays the average speed on the x-axis with corresponding acceleration on the y-axis. Such a map inspired us to evaluate the ability of our model to generate paths which share the same driving style for each driver, and we quantify this by looking at the v-a levels of generated paths.

This kind of data is worth studying since the time dependencies are strong. Indeed, the speed at time t highly depends on the speeds at time $t-1, \dots, t-h$ with h up to more than 100 (seconds).

Figure 5.18 shows the joint distribution between the speed (km/h) and the acceleration (m/s^{-2}), computed over the 200 trajectories of one driver, which somehow characterizes the driver's style. We can see that our model gives a better estimation of such a distribution, which is consistent with the associated MMD^2 score in Table 5.1. On the other hand, Figures 5.19 and 5.20 show the ACF of speed and acceleration. Our model outperforms the RGAN, particularly on the acceleration.

5.6.4 Training details

In this Section, we detail the framework setup we used to train our model. Note that for the sake of fairness when comparing our model to RGAN, we used exactly the same datasets for training, validating and testing, as well as exactly the same sequence length for both models.

For simulated data, we fixed the dataset size for training at 5000 independant rows of sequences of length 10. In all simulated examples, we used the same hyperparameters for our architecture. The simulated examples discussed above were trained using 5 1D residual blocks, with a causal convolutional input layer of length $k = 7$ (green block in

Figure 5.3), and masked convolutions of type B of length $k = 3$ (blue blocks in Figure 5.3), the stride was fixed to 1 and padding to $(k-1)/2$ in both cases. The feature map size was fixed at $h = 256$ in all examples. We implemented these experiments by using PyTorch [Paszke et al., 2017]. The training time is reasonable, taking a few minutes using one NVIDIA TITAN V. The inherent structure of our model imposes the generation step to be sequential (i.e. each prediction needs to be fed back to the input), leading to potentially very long generation step, particularly when the wanted sample length is long, but in all our cases, it does not excess half an hour.

The choice of the prior distribution of τ has been challenged and is promising to estimate heavy-tailed distributions. Figure 5.21 shows the influence of the prior distribution of τ on the financial dataset, for a uniform and a beta prior. The beta prior gives a significantly better estimation of the shape of the marginal distribution. This change of distribution for τ does not have a significant impact on the quality of the autocorrelation's estimation.

For high dimensional time series, we implemented F and G as multi-layer perceptrons (MLP), and we cross-validated the value of the latent space dimension. In this case, the optimal value for $dim(\mathcal{Z})$ is set to 1, which is to be expected since $(X_t)_{t\in\mathbb{Z}}$ is defined along a unidimensional underlying autoregressive AR(1) process. Note that G is not able to reconstruct the error term ϵ_i of Equation 5.6.1. Thus, to correctly rebuild the support of the time series, we bootstrapted the empirical error between $(X_t)_{t\in\mathbb{Z}}$ and the generated sample.

5.7 Discussion

In this paper, we proposed an AIQN architecture designed for time series generation. This method has proven to give excellent results on the MMD^2 score as well as on the ACF. The choice of the beta distribution as a prior helps the model to better fit the marginal distributions, especially when the distributions are heavy-tailed. Moreover, we proposed an appropriate embedding to deal with high dimensional time series which gives accurate results on the ACF and scatterplots.

The proposed method could be extended to non-stationary time series by adding an additional mechanism inside the residual blocks, whose aim would be to learn a potential trend and/or seasonality in the time series. It can also be extended to

asynchronous time series by either considering an adjusted data representation of the input or by adding an extra input layer which aims at projecting potentially incomplete sequences directly in the feature map.

Appendices

This section contains additionnal illustrations about the simulations we performed. We study the behavior of the RGAN and AIQN algorithms along four additional simulated experiments: an autoregressive model AR(1) with a positive coefficient and a Student noise, the GARCH model and two distinct configurations of the Vector Auto-Regressive (VAR) model.

Autoregressive model with Student noise

We simulated an auto regressive stochastic process (AR(1)) as defined in Equation 5.3.3, except that the noise term $(\varepsilon_t)_{t\in\mathbb{Z}}$ follows a Student distribution with parameter $\nu = 4$ (degrees of freedom), meaning that the stochastic process is heavy tailed. For this experiment, we fixed the beta prior's parameters to $\alpha = \beta = 1$ (uniform distribution), and we compare our approach with the RGAN algorithm.
Figures 5.22 and 5.23 show respectively the ACF and the Q-Q plots of the generated samples. We see that RGAN, althought it slighlty underestimates the auto correlations, gives a better estimation of the ACF, and gives a better quantile alignment.

GARCH model

We challenge our approach and the RGAN model to estimate a GARCH(1,1) stochastic process. The parameters of the GARCH model are fixed to $\alpha_0 = .2$, $\alpha_1 = .5$ and $\beta_1 = .3$. Recall that the GARCH model can be interpreted as a white noise of which squared values has auto-correlations. Hence, we expect each model to estimate the auto-correlations on the squared values of $(X_t)_{t\in\mathbb{Z}}$. Figures 5.24 and 5.25 show respectively the ACF (computed on squared data) and the Q-Q plots of the real and the generated sample (generated from AIQN and form RGAN). We see that both models give a good estimation of the quantiles, however the RGAN gives better estimations of the autocorrelations on the squared data.

Vector Auto-Regressive model

In this experiment, we challenge our approach and the RadialGAN to estimate the distribution of a Vector Auto-Regressive (VAR) model. The VAR model of order p and dimension d is the multivariate analogue of the AR(p) model, and is defined by the following recursion:

$$
\begin{aligned}
X_{1,t} &= a^1_{1,1}X_{1,t-1} + \cdots + a^1_{1,d}X_{d,t-1} + \cdots + a^p_{1,1}X_{1,t-p} + \cdots + a^p_{1,d}X_{d,t-p} + \varepsilon_{1,t} \\
X_{2,t} &= a^1_{2,1}X_{1,t-1} + \cdots + a^1_{2,d}X_{d,t-1} + \cdots + a^p_{2,1}X_{1,t-p} + \cdots + a^p_{2,d}X_{d,t-p} + \varepsilon_{2,t} \\
&\vdots \\
X_{d,t} &= a^1_{d,1}X_{1,t-1} + \cdots + a^1_{d,d}X_{d,t-1} + \cdots + a^p_{d,1}X_{1,t-p} + \cdots + a^p_{d,d}X_{d,t-p} + \varepsilon_{d,t}.
\end{aligned}
$$

The parameters of the model defined above are the elements of the matrix $A := (a_{ij})_{i,j\in\{1,\dots,d\}}$, where $a_{ij} \in \mathbb{R}$. In this experiment, we illustrate our method on two distinct configurations of the matrix A, and we set $d = 3$ and $p = 1$. In both configurations, ε_t is a multivariate normal distribution of mean $\mu = (0, 0, 0)$ and identity correlation matrix.

First configuration of the VAR(1)

The first configuration corresponds to the following matrix:

$$
A = \begin{pmatrix} .5 & .2 & .2 \\ .2 & .5 & .2 \\ .2 & .2 & .5 \end{pmatrix}. \tag{5.7.1}
$$

Figures 5.26, 5.27 and 5.28 show the ACF of the real and generated samples, on the first, second and third dimension of $(X_t)_{t\in\mathbb{Z}}$ respectively. We see that for each dimension, our method gives a better estimation of the auto-correlations.

Figures 5.29, 5.30 and 5.31 show the CACF of the real and generated samples. The CACF of Figure 5.29 measures the auto-correlation between the first dimension and the lagged values of the second dimension, which we denote $CACF_{0,1}$. Figures 5.30 and 5.31 respectvely show $CACF_{0,2}$ and $CACF_{1,2}$. We see that in each case, our method gives a better estimation of the cross auto-correlations between each dimensions.
Note that since A is symmetric, we have $CACF_{ij} = CACF_{ji}$.

On the other hand, Figures 5.32, 5.33 and 5.34 show the Q-Q plots between the real and generated samples for each dimension respectively. We see that RGAN gives a better quantile alignment.

Second configuration of the VAR(1)

The second configuration corresponds to the following matrix:

$$A = \begin{pmatrix} .5 & -.2 & -.2 \\ .2 & .5 & -.2 \\ .2 & .2 & .5 \end{pmatrix}. \tag{5.7.2}$$

Figures 5.35, 5.37 and 5.37 show the ACF of the real and generated samples, on the first, second and third dimension of $(X_t)_{t\in\mathbb{Z}}$ respectively. We see that for each dimension, our method gives a better estimation of the auto-correlations.

Figures 5.38, 5.39 and 5.40 show the CACF of the real and generated samples. The CACF of Figure 5.38 measures the auto-correlation between the first dimension and the lagged values of the second dimension, which we denote $CACF_{0,1}$. Figures 5.39 and 5.40 respectvely show $CACF_{0,2}$ and $CACF_{1,2}$. We see that in each case, our method globally gives a better estimation of the cross auto-correlations between each dimensions.
We do not give the values of $CACF_{1,0}$, $CACF_{2,0}$ and $CACF_{2,1}$, but we observed equivalent results.

On the other hand, Figures 5.41, 5.42 and 5.43 show the Q-Q plots between the real and generated samples for each dimension respectively. We see that RGAN gives a better quantile alignment.

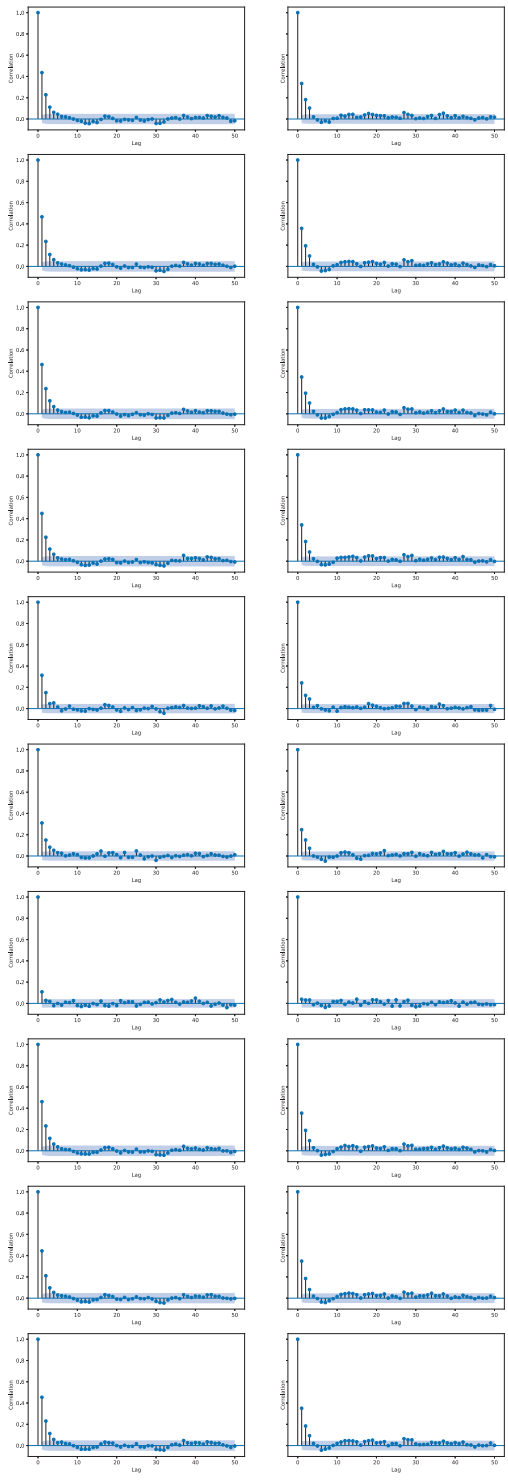

Fig. 5.12 ACF of the real sample (left) and ACF of the generated sample (right) using our proposed method (first ten dimensions). The underlying stochastic process is of dimension $d = 200$ and is defined by Equation 5.6.1. We see that our model is able to accurately estimate the auto correlations of most of the dimensions.

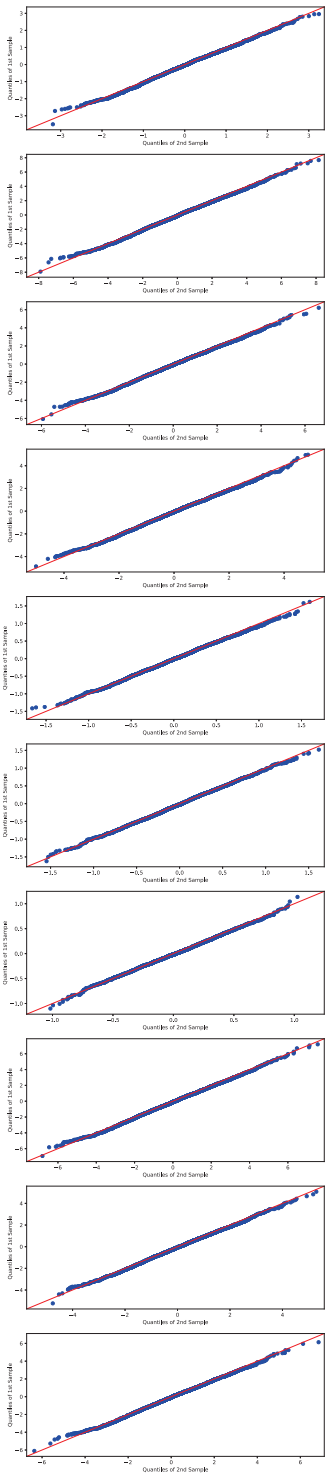

Fig. 5.13 Scatterplots of the real sample (left) and scatterplot of the generated sample (right) using our proposed method (first ten dimensions). The underlying stochastic process is of dimension $d = 200$ and is defined by Equation 5.6.1. We see that our model is able to accurately estimate the marginal distribution of most of the dimensions.

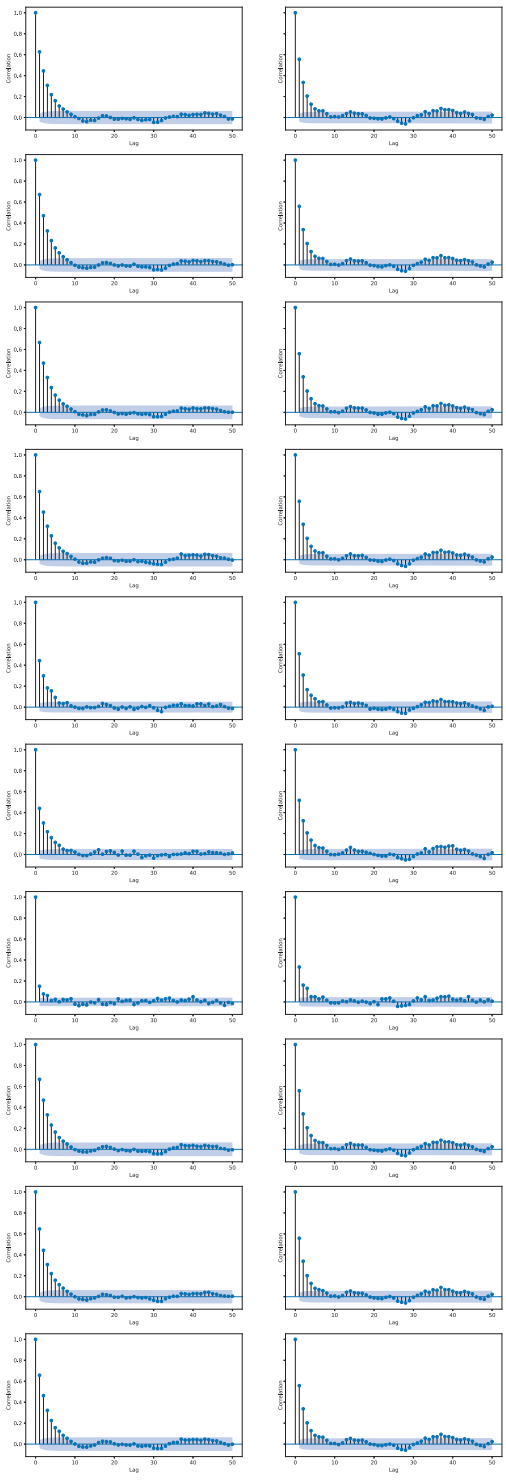

Fig. 5.14 ACF of the real sample (left) and ACF of the generated sample (right) using our proposed method (first ten dimensions). The underlying stochastic process is of dimension $d = 200$ and is defined by Equation 5.6.2 and is heteroskedastic. The variance depends on the values of the underlying AR(1) stochastic process. We see that our model is able to accurately estimate the auto correlations of most of the dimensions.

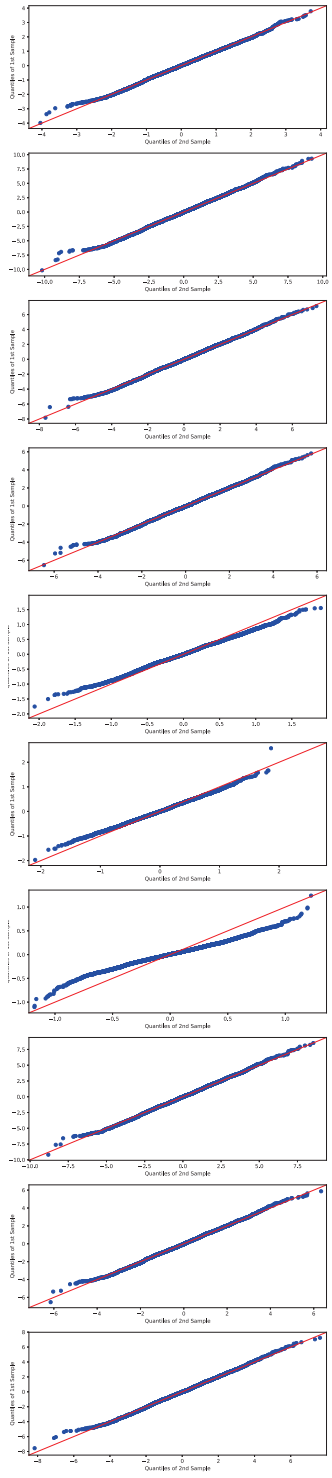

Fig. 5.15 Scatterplots of the real sample (left) and scatterplots of the generated sample (right) using our proposed method (first ten dimensions). The underlying stochastic process is of dimension $d = 200$ and is defined by Equation 5.6.2 and is heteroskedastic. The variance depends on the values of the underlying AR(1) stochastic process. We see that our model is able to accurately estimate the marginal distributions of most of the dimensions.

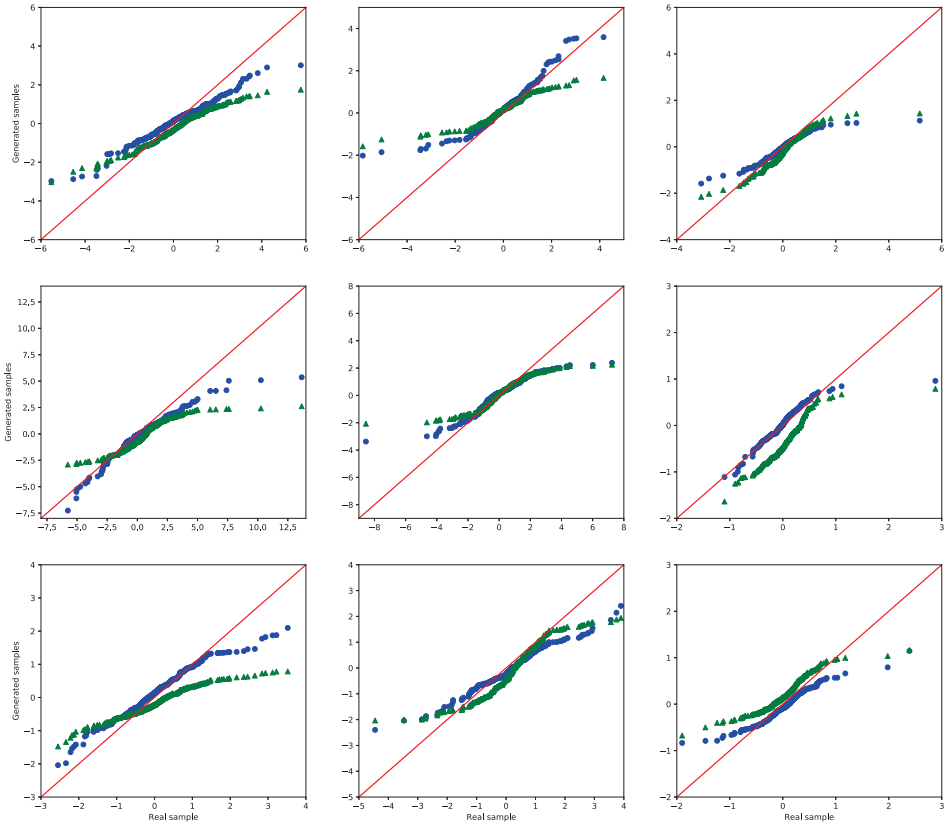

Fig. 5.16 Q-Q plots of the real sample (x-axis) vs the generated samples drawn from AIQN (blue, circle markers) and RGAN (green, triangle markers), on each equity of the financial dataset.

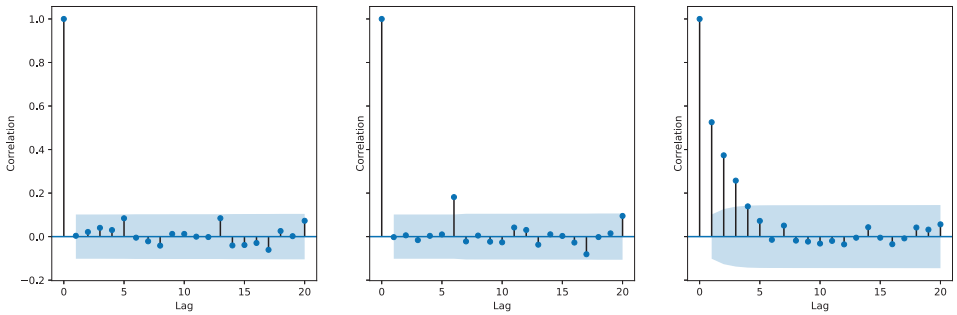

Fig. 5.17 ACF of real (left), AIQN (middle) and GAN (right) computed on the squared increments of the end-of-day prices, on one equity of the financial dataset.

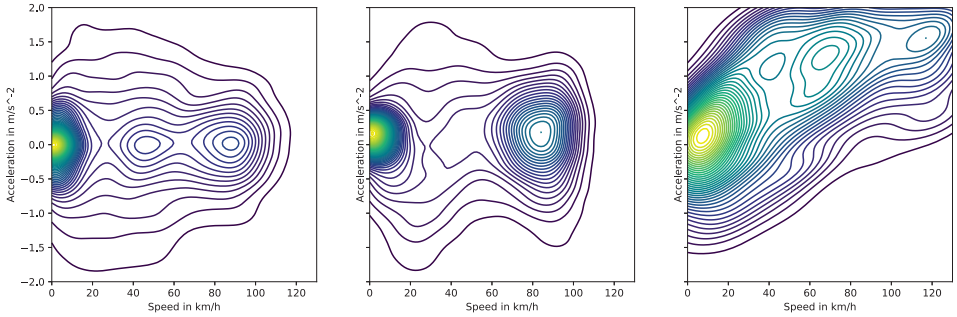

Fig. 5.18 Joint plot of speed and acceleration of the real (left), AIQN (middle) and RGAN (right) samples.

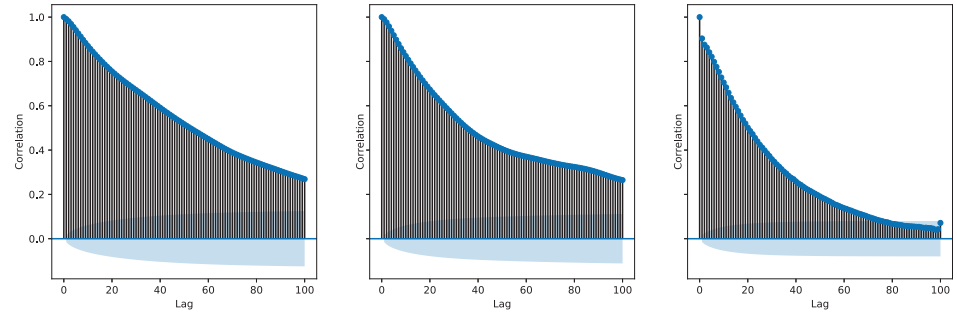

Fig. 5.19 ACF of speed of the real (left), AIQN (middle) and RGAN (right) samples.

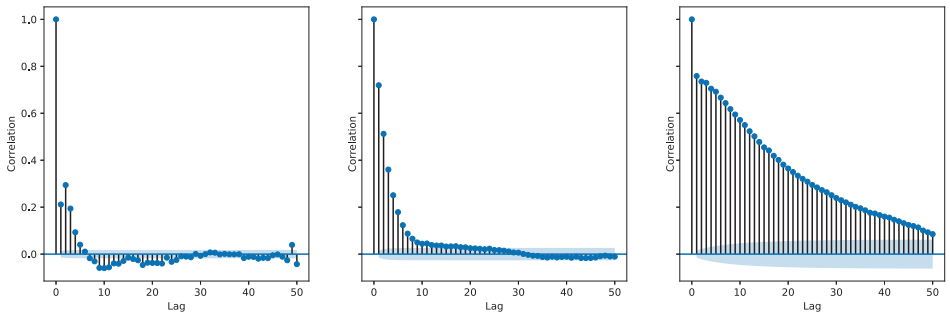

Fig. 5.20 ACF of acceleration of the real (left), AIQN (middle) and RGAN (right) samples.

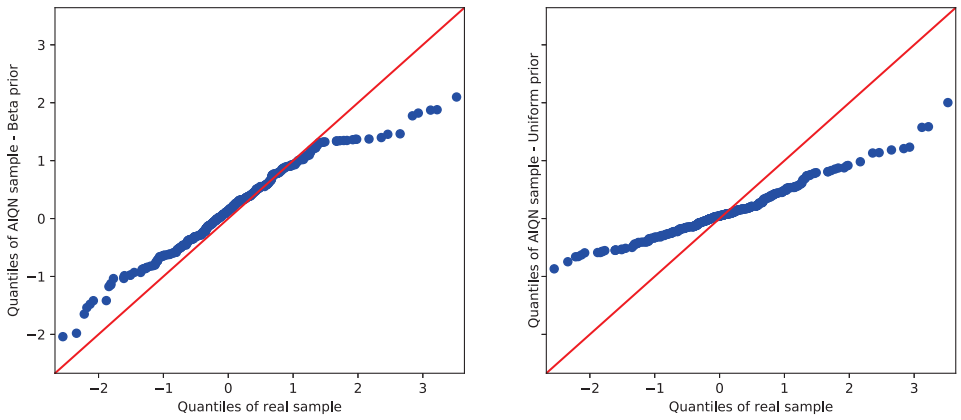

Fig. 5.21 Q-Q plots of the real sample vs the generated sample drawn from AIQN with a beta $\mathcal{B}(1/2, 1/2)$ prior (left) and with a uniform $\mathcal{U}([0, 1])$ prior (right), on one equity of the financial dataset.

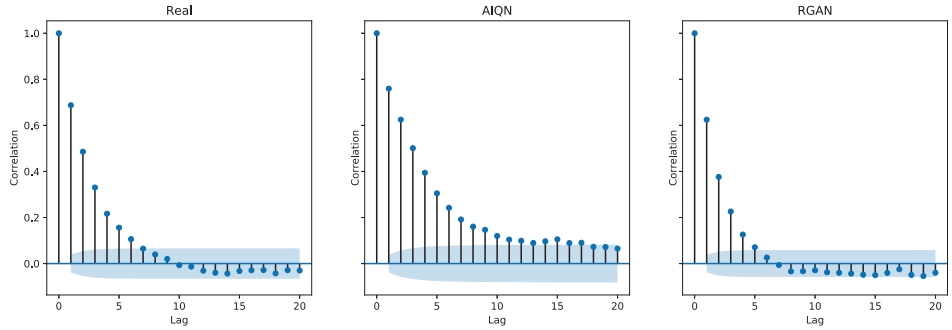

Fig. 5.22 ACF of the real (left), AIQN (middle) and RGAN (right) samples. The underlying stochastic process is an autoregressive AR(1) model with a coefficient of .7. In this experiment, the noise term of the AR(1) model is Student distributed with 4 degrees of freedom, such that the model is heavy tailed. RGAN gives a globally better estimation of the auto correlations.

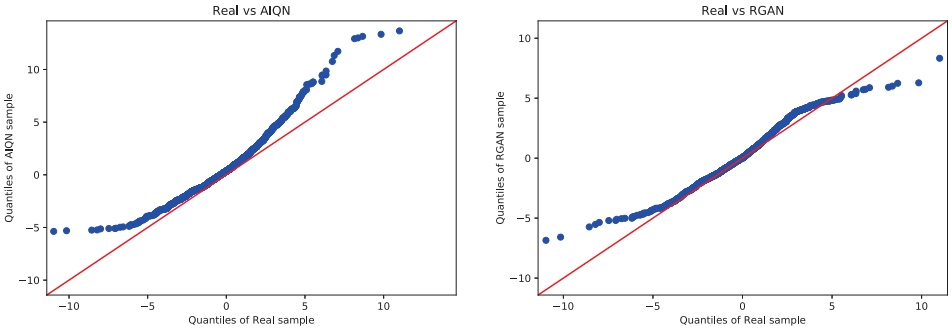

Fig. 5.23 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and the RGAN sample (y-axis, right). The underlying stochastic process is an autoregressive AR(1) model with a coefficient of .7. In this experiment, the noise term of the AR(1) model is Student distributed with 4 degrees of freedom, such that the model is heavy tailed. RGAN gives a better quantile alignment on this experiment.

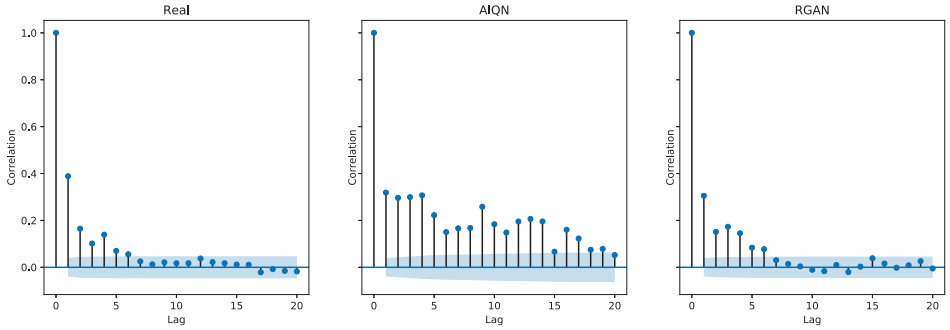

Fig. 5.24 ACF of the real (left), AIQN (middle) and RGAN (right) samples. The ACF is computed on the squared data for this experiment since the underlying stochastic process is a GARCH(1,1). RGAN gives more accurate estimations of the auto-correlations of the squared data.

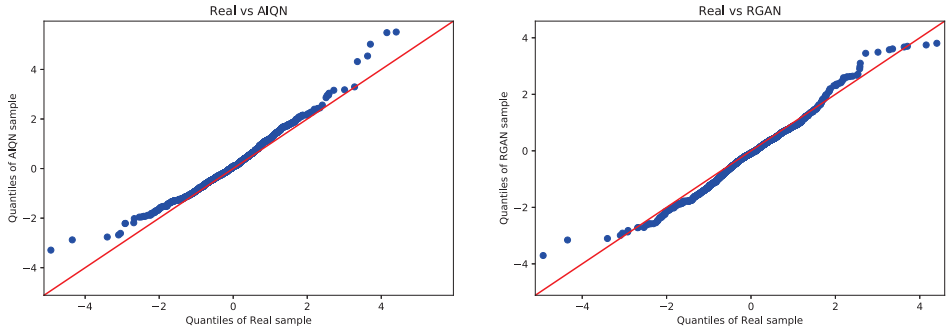

Fig. 5.25 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and the RGAN sample (y-axis, right). The underlying stochastic process is a GARCH(1,1), and we see that both models give an accurate estimation of the quantiles.

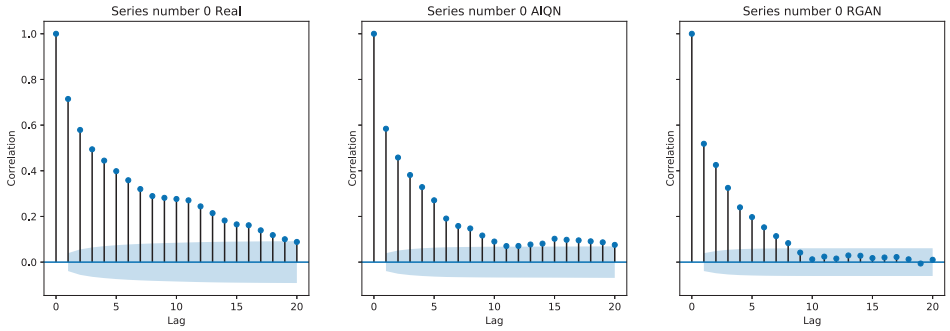

Fig. 5.26 ACF of the real (left), AIQN (middle) and RGAN (right) samples on the first dimension of a Vector Auto-Regressive (VAR) model in the first configuration. We see that our method gives a better estimation of the auto-correlations.

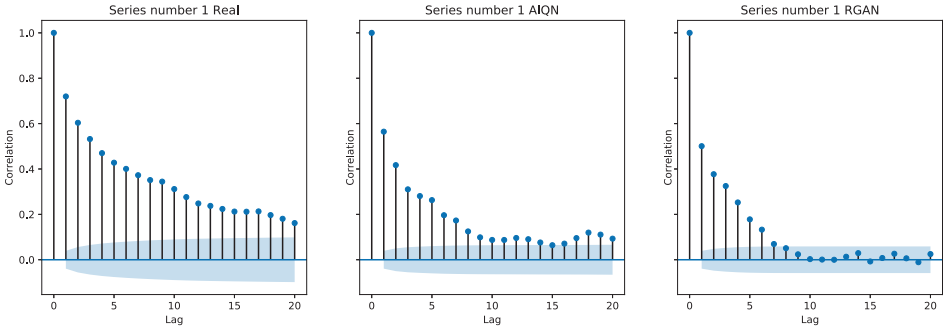

Fig. 5.27 ACF of the real (left), AIQN (middle) and RGAN (right) samples on the second dimension of a Vector Auto-Regressive (VAR) model in the first configuration. We see that our method gives a better estimation of the auto-correlations.

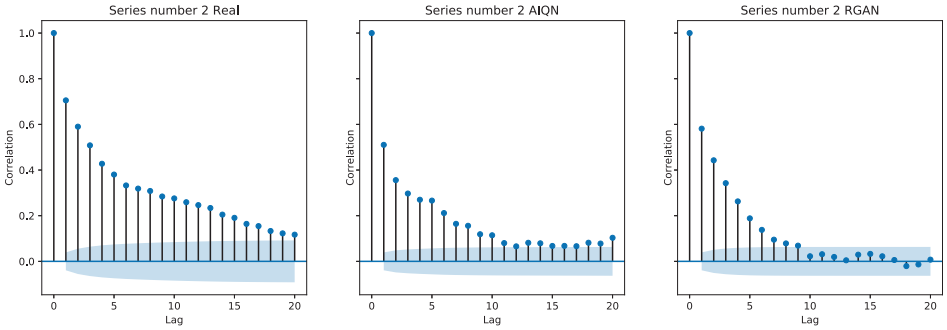

Fig. 5.28 ACF of the real (left), AIQN (middle) and RGAN (right) samples on the third dimension of a Vector Auto-Regressive (VAR) model in the first configuration. We see that our method gives a better estimation of the auto-correlations.

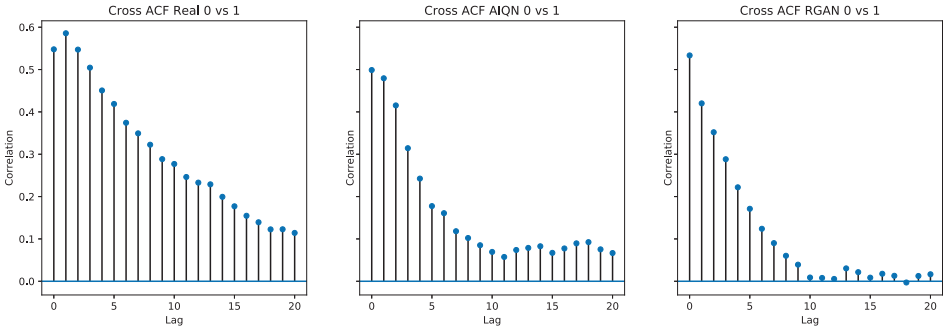

Fig. 5.29 CACF of the real (left), AIQN (middle) and RGAN (right) samples of the first dimension with respect to the second dimension. The underlying stochastic process is a Vector Auto-Regressive (VAR) model in the first configuration. We see that our method gives a better estimation of the auto-correlations.

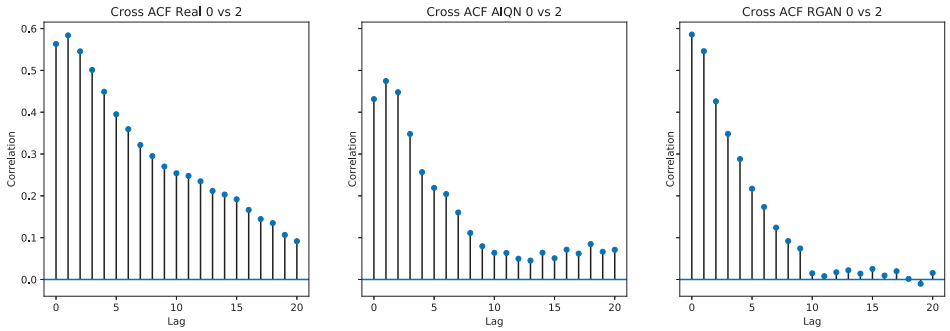

Fig. 5.30 CACF of the real (left), AIQN (middle) and RGAN (right) samples of the first dimension with respect to the third dimension. The underlying stochastic process is a Vector Auto-Regressive (VAR) model in the first configuration. We see that our method gives a better estimation of the auto-correlations.

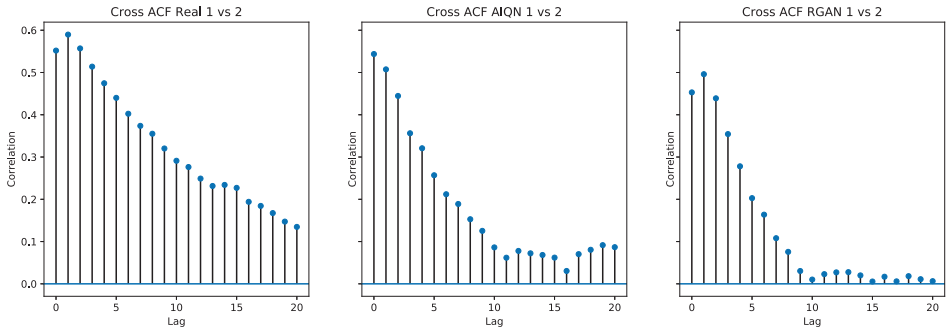

Fig. 5.31 CACF of the real (left), AIQN (middle) and RGAN (right) samples of the second dimension with respect to the third dimension. The underlying stochastic process is a Vector Auto-Regressive (VAR) model in the first configuration. We see that our method gives a better estimation of the auto-correlations.

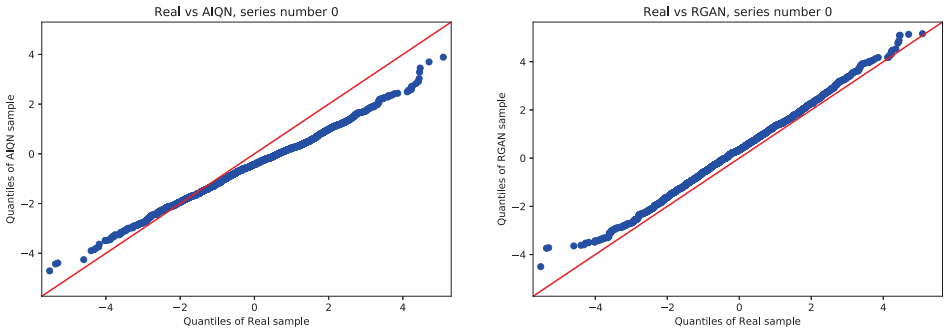

Fig. 5.32 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and RGAN sample (y-axis, right), on the first dimension of a Vector Auto-Regressive (VAR) model in the first configuration. We see that the RGAN gives a better estimation of the support of $(X_t)_{t\in\mathbb{Z}}$.

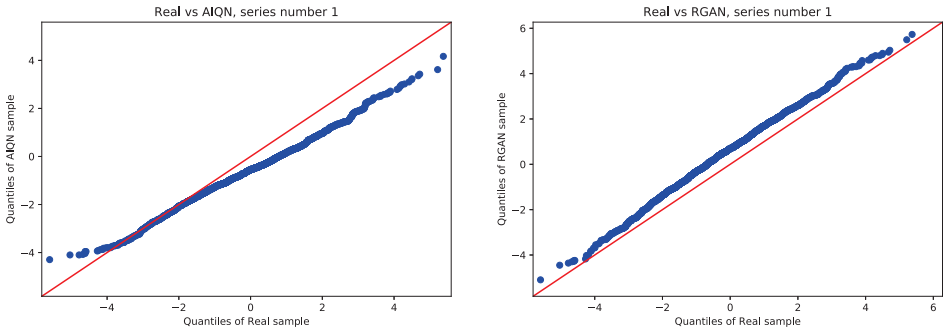

Fig. 5.33 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and RGAN sample (y-axis, right), on the second dimension of a Vector Auto-Regressive (VAR) model in the first configuration. We see that the RGAN gives a better estimation of the support of $(X_t)_{t\in\mathbb{Z}}$.

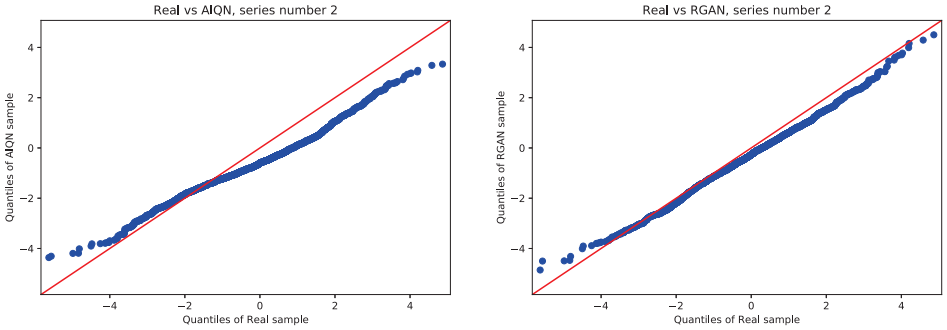

Fig. 5.34 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and RGAN sample (y-axis, right), on the third dimension of a Vector Auto-Regressive (VAR) model in the first configuration. We see that the RGAN gives a better estimation of the support of $(X_t)_{t\in\mathbb{Z}}$.

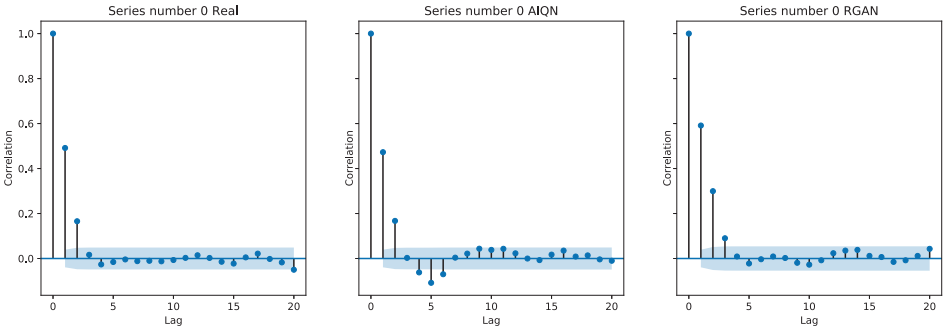

Fig. 5.35 ACF of the real (left), AIQN (middle) and RGAN (right) samples on the first dimension of a Vector Auto-Regressive (VAR) model in the second configuration. We see that our method gives a better estimation of the auto-correlations.

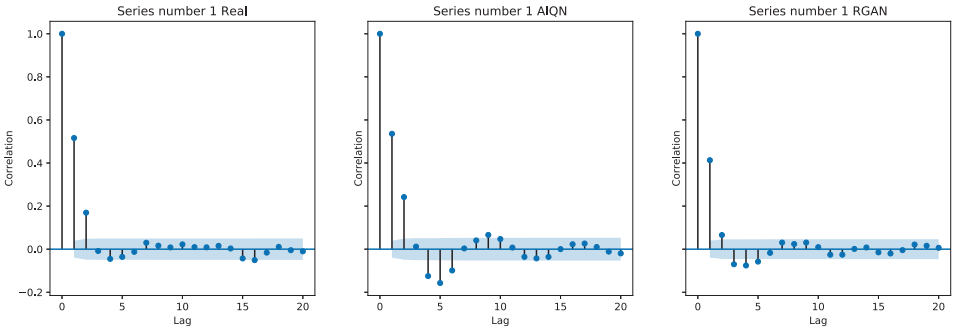

Fig. 5.36 ACF of the real (left), AIQN (middle) and RGAN (right) samples on the second dimension of a Vector Auto-Regressive (VAR) model in the second configuration. We see that our method gives a better estimation of the auto-correlations.

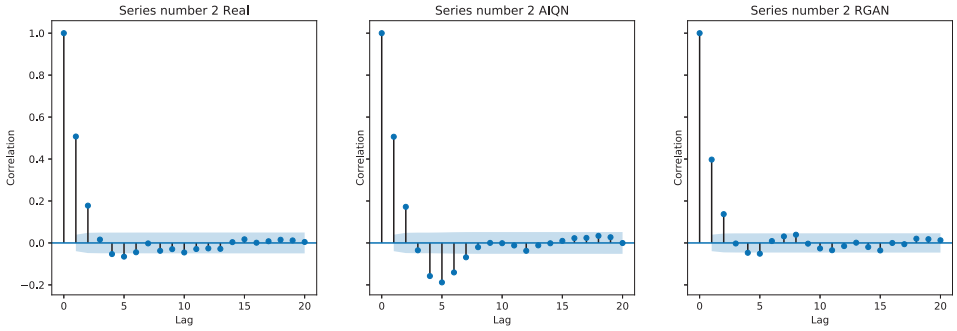

Fig. 5.37 ACF of the real (left), AIQN (middle) and RGAN (right) samples on the third dimension of a Vector Auto-Regressive (VAR) model in the second configuration. We see that our method gives a better estimation of the auto-correlations.

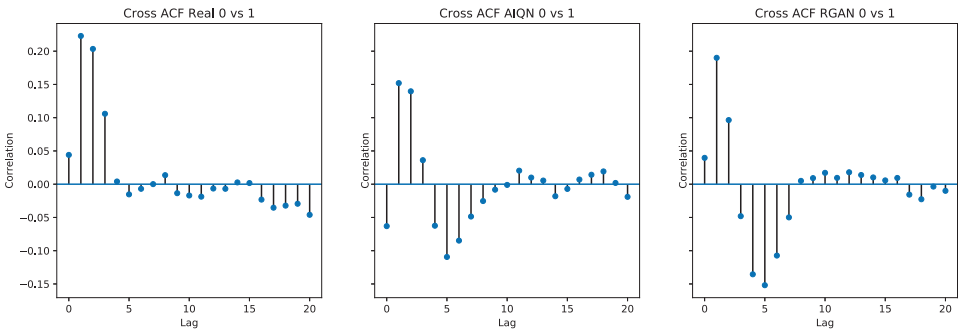

Fig. 5.38 CACF of the real (left), AIQN (middle) and RGAN (right) samples of the first dimension with respect to the second dimension. The underlying stochastic process is a Vector Auto-Regressive (VAR) model in the second configuration. We see that our method gives a better estimation of the auto-correlations.

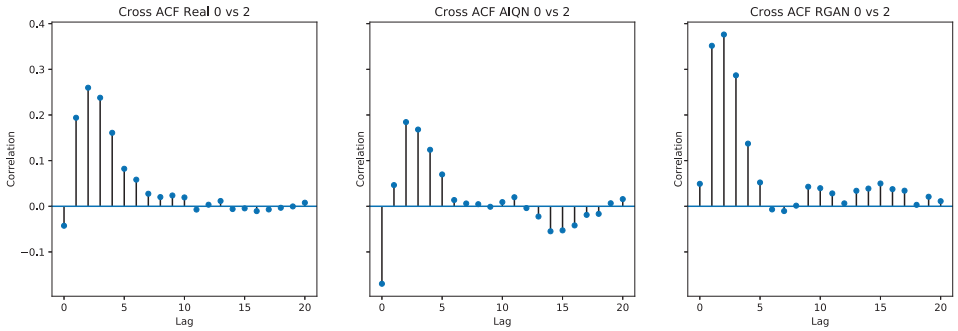

Fig. 5.39 CACF of the real (left), AIQN (middle) and RGAN (right) samples of the first dimension with respect to the third dimension. The underlying stochastic process is a Vector Auto-Regressive (VAR) model in the second configuration. We see that our method gives a better estimation of the auto-correlations.

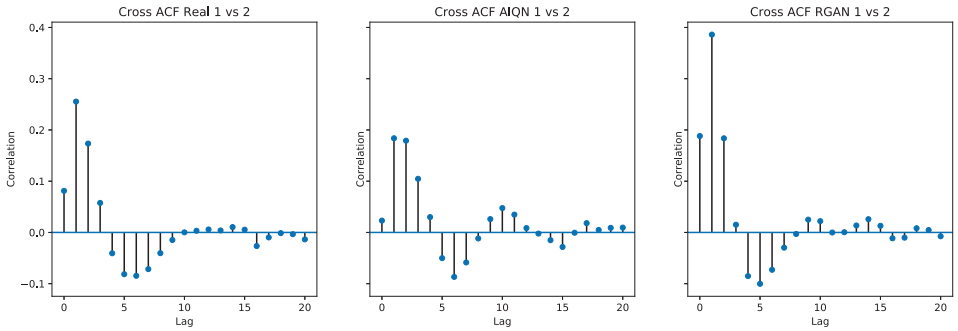

Fig. 5.40 CACF of the real (left), AIQN (middle) and RGAN (right) samples of the second dimension with respect to the third dimension. The underlying stochastic process is a Vector Auto-Regressive (VAR) model in the second configuration. We see that our method gives a better estimation of the auto-correlations.

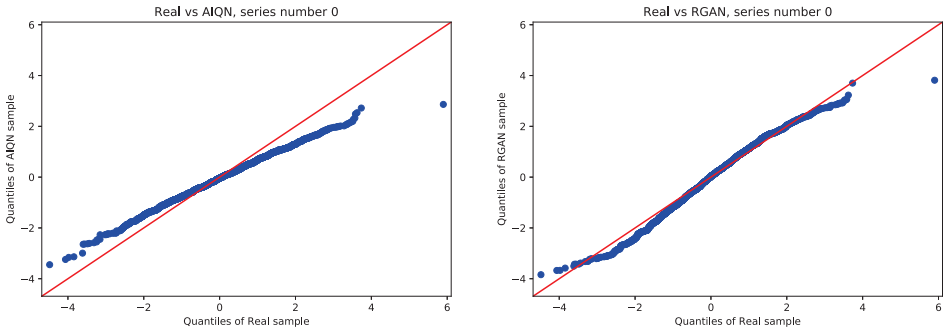

Fig. 5.41 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and RGAN sample (y-axis, right), on the first dimension of a Vector Auto-Regressive (VAR) model in the second configuration. We see that the RGAN gives a better estimation of the support of $(X_t)_{t\in\mathbb{Z}}$.

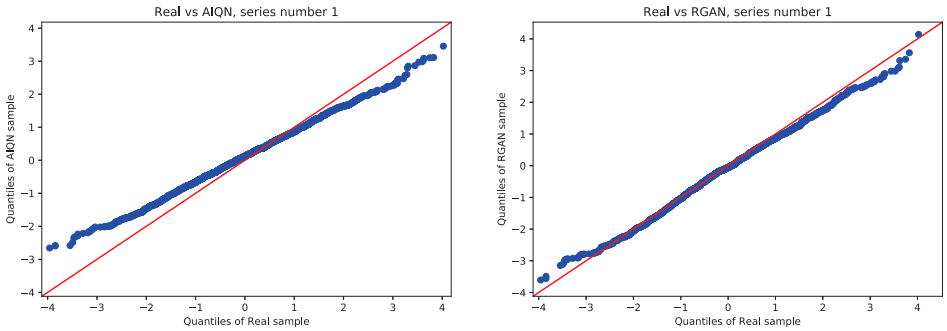

Fig. 5.42 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and RGAN sample (y-axis, right), on the second dimension of a Vector Auto-Regressive (VAR) model in the second configuration. We see that the RGAN gives a better estimation of the support of $(X_t)_{t\in\mathbb{Z}}$.

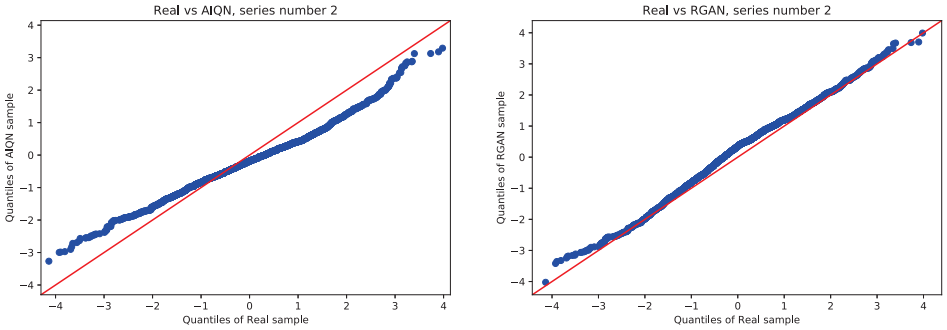

Fig. 5.43 Q-Q plots of the real sample (x-axis) vs the AIQN sample (y-axis, left) and RGAN sample (y-axis, right), on the third dimension of a Vector Auto-Regressive (VAR) model in the second configuration. We see that the RGAN gives a better estimation of the support of $(X_t)_{t\in\mathbb{Z}}$.

Conclusion

Summary of contributions and main ideas

This thesis focuses on the use of modern machine learning methods in presence of incomplete data with applications in insurance. The litterature of machine learning dealing with incomplete data is scarce. Moreover, many deep learning works focus on working with data such as images, text, video or audio data, but few works have been done dealing with tabular data and time series (more generally, data which needs statistical arguments to assess the quality of a model).

In Chapter 3, we focused on building a prediction model to predict a highly censored time-related random variable. Very few works including [Lopez et al., 2016, Gerber et al., 2018, Wüthrich, 2018] address such a task with machine learning. The usual method (chain ladder) for such a task makes strong assumtions on the structure of the data and on the censoring scheme. In this contribution, we showed that we are able to give more accurate predictions than the chain ladder.
In Chapter 4, we proposed a neural architecture to combine multiple datasets to build a better predictive model on each dataset. The usual deep learning tools to perform a domain transfer are applied to image-to-image translation [Zhu et al., 2017a, Choi et al., 2018, Kim et al., 2017] (among other applications), but very few has been proposed on tabular data [Yoon et al., 2018b]. We showed that our proposed approach outperforms the state-of-the-art model for domain transfer applied on tabular data.
In Chapter 5, we proposed an auto-regressive model to accurately estimate the conditional distribution of a time series. Deep learning works on time series are rare, and most of the contributions focus on predicting future values of a time series. In this paper, we showed that we give a better estimation of the support of the time series and of its auto-correlations (and cross auto-correlations) than the state-of-the-art model. We also proposed an adaptation of our approach to address the case when the underlying stochastic process is heavy tailed and high dimensional.

From all the contributions presented in this thesis, the takeaway message is that developping methods to perform advanced tasks on tabular data (e.g. estimating the distribution of a random vector, identifying mixtures in a multivariate distribution or transporting a measure to an other) is challenging. Indeed, the model's quality can be easily assessed when addressing a prediction task, but requires more complex tools on other tasks such as distribution estimation, which is not addressed in image processing or text analysis.

Perspectives of future contributions

This thesis is far form tackling every data incompleteness scheme, and we identified some future potential contributions:

- **Missing values imputation.** Missing values are almost omnipresent in practice, and are a natural extension of this thesis. As evocated in Chapter 2, deep learning works tackling missing values imputation are scarce, either focus on a particular missingness mechanism or on filling missing entries to address a prediction task, and lack of statistical arguments to assess the imputation's quality. No thorough work on multiple imputation using deep learning methods has been proposed, and yet such approaches have a lot of potential to identify the true underlying distribution of a dataset.

- **Going deeper with cross-domain modelling.** Chapter 4 proposed to perform a domain transfer between multiple datasets in order to transform each dataset to another without losing predictive information. Working with latent representations of multivariate domains is a field of research which is getting more and more popular [Lample et al., 2017, Chen and Denoyer, 2017, Xu and Veeramachaneni, 2018, Yang et al., 2018], and a natural extension of the work we proposed would be to compute an efficient latent representation of each datasets and train a prediction model directly through the latent space instead of performing a data augmentation. This allows to train a latent model which can be later used when a new dataset enters in the study, without the need to re-train the domain transfer model.

- **Cross-domain modelling for addressing train/test distribution mismatch.** In the classical supervised learning task, the distributions of the train

and test datasets are supposed to be identical. Although being reasonable, this assumption is not always true in practice, particularly when the train and test datasets are time-dependant (e.g. the train dataset corresponds to observations recorded at year I and the test at year $I + 1$), where change-points can occur. This often leads to a bias in predictions. Hence, the cross-domain modelling introduced in Chapter 4 opens the way to equalize the distributions of the train and test sets, then training an unbiased prediction model on the unified datasets.

- **Non stationary time series.** The proposed method in Chapter 5 assumes that the stochastic process is stationnary. An extension of this work is to propose an architecture to modellize a non-stationnary time series, by dynamically detect trend and seasonality factors while estimating the auto-correlation structure on raw and squared data at the same time. This needs to train separate models, each designed to detect one property of the time series, and combine their intermediate representations.

- **Asynchronous time series.** Another strong assumption we made in Chapter 5 besides the stationarity was that the records of the time series were synchronous, i.e. all dimensions of the time series were recorded simultaneously and at a regular time interval. Works on asynchronous time series prediction have been proposed [Binkowski et al., 2017, Che et al., 2018], but to the best of our knowledge, no work has been proposed to estimate the distribution of an asynchronous stochastic process.

References

[Anderson, 1980] Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *Technical report*.

[Antonio and Plat, 2014] Antonio, K. and Plat, R. (2014). Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal*, 2014(7):649–669.

[Arjas, 1989] Arjas, E. (1989). The claims reserving problem in non-life insurance: Some structural ideas. *ASTIN Bulletin*, 19.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223.

[Badescu et al., 2016] Badescu, A. L., Lin, X. S., and Tang, D. (2016). A marked cox model for the number of ibnr claims: Theory. *Insurance: Mathematics and Economics*, 69(C):29–37.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

[Binkowski et al., 2017] Binkowski, M., Marti, G., and Donnat, P. (2017). Autoregressive convolutional neural networks for asynchronous time series. *CoRR*, abs/1703.04122.

[Bojanowski et al., 2016] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

[Bollerslev, 1986] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327.

[Borovykh et al., 2017] Borovykh, A., Bohte, S., and Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

[Box and Jenkins, 1970] Box, G. and Jenkins, G. (1970). Time series analysis: Forecasting and control.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Chapman and Hall.

[Brockwell and Davis, 1991] Brockwell, P. J. and Davis, R. A. (1991). *Time Series: Theory and Methods*. Springer Series in Statistics.

[Brownlees et al., 2015] Brownlees, C., Joly, E., Lugosi, G., et al. (2015). Empirical risk minimization for heavy-tailed losses. *The Annals of Statistics*, 43(6):2507–2536.

[Che et al., 2018] Che, Z., Purushotham, S., Li, G., Jiang, B., and Liu, Y. (2018). Hierarchical deep generative models for multi-rate multivariate time series. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 784–793. PMLR.

[Chen and Denoyer, 2017] Chen, M. and Denoyer, L. (2017). Multi-view generative adversarial networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 175–188. Springer.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. pages 785–794.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

[Choi et al., 2018] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. (2018). Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th international conference on Machine learning*.

[Connor et al., 1994] Connor, J. T., Martin, R. D., and Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254.

[Cox, 1972] Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[Dabney et al., 2018a] Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018a). Implicit quantile networks for distributional reinforcement learning. *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

[Dabney et al., 2018b] Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018b). Distributional reinforcement learning with quantile regression. *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database.

[Denning, 1987] Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. abs/1810.04805.

[Du et al., 2017] Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298. ACM.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[Dumoulin et al., 2018] Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. (2018). Feature-wise transformations. *Distill*. https://distill.pub/2018/feature-wise-transformations.

[Dumoulin et al., 2016] Dumoulin, V., Shlens, J., and Kudlur, M. (2016). A learned representation for artistic style.

[E. King, 2009] E. King, D. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758.

[Engle, 1982] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007.

[Esteban et al., 2017] Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

[Evgeniou and Pontil, 2004] Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. pages 109–117.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

[Gatys et al., 2015a] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015a). A neural algorithm of artistic style.

[Gatys et al., 2015b] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015b). Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In *Bernstein Conference 2015*, pages 219–219.

[Gerber et al., 2018] Gerber, G., Le Faou, Y., Lopez, O., and Trupin, M. (2018). The impact of churn on client value in health insurance, evaluation using a random forest under random censoring. working paper or preprint.

[Germain et al., 2015] Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42.

[Ghiasi et al., 2017] Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., and Shlens, J. (2017). Exploring the structure of a real-time, arbitrary neural artistic stylization network.

[Giles et al., 2001] Giles, C. L., Lawrence, S., and Tsoi, A. C. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, 44(1-2):161–183.

[Goix et al., 2016] Goix, N., Sabourin, A., and Clémençon, S. (2016). Sparse representation of multivariate extremes with applications to anomaly ranking. In *AISTATS*, pages 75–83.

[Gondara and Wang, 2017] Gondara, L. and Wang, K. (2017). Multiple imputation using deep denoising autoencoders.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. pages 2672–2680.

[Graves, 2013] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

[Gretton et al., 2008] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. J. (2008). A kernel method for the two-sample problem. *CoRR*, abs/0805.2368.

[Guo et al., 2017] Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., and Wang, J. (2017). Long text generation via adversarial training with leaked information. *CoRR*, abs/1709.08624.

[Haastrup and Arjas, 1996] Haastrup, S. and Arjas, E. (1996). Claims reserving in continuous time; a nonparametric bayesian approach. *ASTIN Bulletin*, 26(2):139–164.

[Hamilton, 1994] Hamilton, J. D. (1994). Time series analysis. *Princeton university press*, 2.

[Hastie and Tibshirani, 1986] Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statist. Sci.*, 1(3):297–310.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc.

[Heskes, 2000] Heskes, T. (2000). Empirical bayes for learning to learn.

[Heusel et al., 2017] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500.

[Hiabu et al., 2016] Hiabu, M., Margraf, C., Martínez-Miranda, M. D., and Nielsen, J. P. (2016). The link between classical reserving and granular reserving through double chain ladder and its extensions. *British Actuarial Journal*, 21(1):97–116.

[Hinton, 2012] Hinton, G. (2012). Neural networks for machine learning. *Coursera, video lectures*.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Huang and Belongie, 2017] Huang, X. and Belongie, S. J. (2017). Arbitrary style transfer in real-time with adaptive instance normalization.

[Isola et al., 2016] Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *arXiv preprint*.

[J. Huber, 1964] J. Huber, P. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:73–101.

[Jalalzai et al., 2018] Jalalzai, H., Clémençon, S., and Sabourin, A. (2018). On binary classification in extreme regions. In *Advances in Neural Information Processing Systems*, pages 3092–3100.

[Javaid et al., 2016] Javaid, A., Niyaz, Q., Sun, W., and Alam, M. (2016). A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26.

[Jin, 2013] Jin, X. (2013). Micro-level loss reserving models with applications in workers compensation insurance. *University of Wisconsin-Madison*.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[Kamaljot et al., 2015] Kamaljot, S., Ranjeet, K. S., and Dinesh, K. (2015). Comment volume prediction using neural networks and decision trees. *IEEE UKSim-AMSS 17th International Conference on Computer Modelling and Simulation, UKSim2015 (UKSim2015)'*.

[Kaplan and Meier, 1958] Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481.

[Karras et al., 2017] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. pages 3146–3154.

[Kim et al., 2017] Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. (2017). Learning to discover cross-domain relations with generative adversarial networks.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Koenker and Hallock, 2001] Koenker, R. and Hallock, K. (2001). Quantile regression: An introduction. *Journal of Economic Perspectives*, 15(4):43–56.

[Krishnan et al., 2017] Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. pages 2101–2109.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

[Kumar, 1995] Kumar, S. (1995). Classification and detection of computer intrusions.

[Kuo, 2018] Kuo, K. (2018). Deeptriangle: A deep learning approach to loss reserving. *arXiv preprint arXiv:1804.09253*.

[Lample and Conneau, 2019] Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

[Lample et al., 2018] Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. *CoRR*, abs/1804.07755.

[Lample et al., 2017] Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., and Ranzato, M. (2017). Fader networks: Manipulating images by sliding attributes. abs/1706.00409.

[Lane and Brodley, 1997] Lane, T. and Brodley, C. E. (1997). An application of machine learning to anomaly detection. In *Proceedings of the 20th National Information Systems Security Conference*, volume 377, pages 366–380. Baltimore, USA.

[Larsen, 2007] Larsen, C. R. (2007). An individual claims reserving model. *ASTIN Bulletin*, 37(1):113–132.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun et al., 1989] LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, pages 143–155.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. 2(3):4.

[Li et al., 2017] Li, Y., Liu, S., Yang, J., and Yang, M.-H. (2017). Generative face completion. pages 3911–3919.

[Liao et al., 2005] Liao, X., Xue, Y., and Carin, L. (2005). Logistic regression with an auxiliary data source. pages 505–512.

[Little and Rubin, 2019] Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

[Lopez et al., 2016] Lopez, O., Milhaud, X., Thérond, P.-E., et al. (2016). Tree-based censored regression with applications in insurance. *Electronic journal of statistics*, 10(2):2685–2716.

[Lunt, 1990] Lunt, T. (1990). Ides: An intelligent system for detecting intruders. *Proceedings of the Symposium: Computer Security, Threat and Countermeasures.*

[Mahendran and Vedaldi, 2015] Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196.

[McCann et al., 2017] McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association.*

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Nesterov, 1983] Nesterov, Y. (1983). A method for solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376.

[Norberg, 1993] Norberg, R. (1993). Prediction of outstanding liabilities in non-life insurance. *Insurance Mathematics and Economics*, 13.

[Norberg, 1999] Norberg, R. (1999). Prediction of outstanding liabilities ii. model variations and extensions. *ASTIN Bulletin*, 29(1):5–25.

[Oord et al., 2016] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759.*

[Ostrovski et al., 2018] Ostrovski, G., Dabney, W., and Munos, R. (2018). Autoregressive quantile networks for generative modeling. *Proceedings of the 35th International Conference on Machine Learning*.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Perez et al., 2018] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Peyré et al., 2019] Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

[Pigeon et al., 2013] Pigeon, M., Antonio, K., and Denuit, M. (2013). Individual loss reserving with the multivariate skew normal framework. *ASTIN Bulletin*, 43(3):399–428.

[Pu et al., 2016] Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[Robinson, 1983] Robinson, P. M. (1983). Nonparametric estimators for time series. *Journal of Time Series Analysis*, 4(3):185–207.

[Roos et al., 2006] Roos, T., Grünwald, P., Myllymäki, P., and Tirri, H. (2006). Generalization to unseen cases. In *Advances in neural information processing systems*, pages 1129–1136.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rubin, 1976] Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

[Rubin, 2004] Rubin, D. B. (2004). *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons.

[Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Salimans et al., 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

[Shon and Moon, 2007] Shon, T. and Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177:3799–3821.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Socher et al., 2010] Socher, R., Manning, C. D., and Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. *In Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

[Stekhoven and Bühlmann, 2011] Stekhoven, D. J. and Bühlmann, P. (2011). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

[Sutskever et al., 2011] Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*.

[Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

[Taylor et al., 2008] Taylor, G., McGuire, G., and Sullivan, J. (2008). Individual claim loss reserving conditioned by case estimates. *Annals of Actuarial Science*, 3.

[Tsybakov, 1986] Tsybakov, A. B. (1986). Robust reconstruction of functions by the local-approximation method. *Problemy Peredachi Informatsii*, 22(2):69–84.

[van Buuren and Groothuis-Oudshoorn, 2011] van Buuren, S. and Groothuis-Oudshoorn, C. (2011). Mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45.

[van den Oord et al., 2016a] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499.

[van den Oord et al., 2016b] van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016b). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798.

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg.

[Verbelen et al., 2017] Verbelen, R., Antonio, K., Claeskens, G., and Crèvecoeur, J. (2017). Predicting daily ibnr claim counts using a regression approach for the occurrence of claims and their reporting delay. *Working paper*.

[Villani, 2009] Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103.

[Wiens et al., 2014] Wiens, J., Guttag, J., and Horvitz, E. (2014). A study in transfer learning: leveraging data from multiple hospitals to enhance hospital-specific predictions. *Journal of the American Medical Informatics Association*, 21(4):699–706.

[Wüthrich, 2018] Wüthrich, M. V. (2018). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018(6):465–480.

[Wüthrich, 2017] Wüthrich, M. (2017). Covariate selection from telematics car driving data. *European Actuarial Journal*, 7.

[Xu et al., 2015] Xu, D., Ricci, E., Yan, Y., Song, J., and Sebe, N. (2015). Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*.

[Xu and Veeramachaneni, 2018] Xu, L. and Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*.

[Yang et al., 2018] Yang, L., Wang, Y., Xiong, X., Yang, J., and Katsaggelos, A. K. (2018). Efficient video object segmentation via network modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6499–6507.

[Yoon et al., 2018a] Yoon, J., Jordon, J., and Van Der Schaar, M. (2018a). Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*.

[Yoon et al., 2018b] Yoon, J., Jordon, J., and van der Schaar, M. (2018b). Radialgan: Leveraging multiple datasets to improve target-specific predictive models using generative adversarial networks.

[Yu et al., 2017] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

[Zhang et al., 2018] Zhang, C., Kuppannagari, S. R., Kannan, R., and Prasanna, V. K. (2018). Generative adversarial network for synthetic time series data generation in smart grids. *IEEE SmartgridComm 2018*.

[Zhu et al., 2017a] Zhu, J., Park, T., Isola, P., and Efros, A. A. (2017a). Unpaired image-to-image translation using cycle-consistent adversarial networks.

[Zhu et al., 2017b] Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. (2017b). Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476.

List of Figures

List of Tables