Model-based versus model-free reinforcement learning in quantitative asset management

David Saltiel

Thèse de Doctorat

Mention : Sciences et Technologies de l'Information et de la Communication
Spécialité: Informatique et applications

présentée à **l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)**

de l'Université du Littoral Côte d'Opale

par

David Saltiel

pour obtenir le grade de Docteur de l'Université du Littoral Côte d'Opale

Model-based versus model-free reinforcement learning in quantitative asset management

Soutenance prévue le mercredi 7 décembre 2022, devant le jury d'examen:

Laetitia Jourdan, Professeure des universités, Université de Lille	**Présidente du jury**
Emmanuel Lepinette, Maitre de conférences, Université PARIS – DAUPHINE	**Rapporteur**
Jonathan Weber, Maitre de conférences, Université de Haute-Alsace	**Rapporteur**
Nicole El Karoui, Professeure émérite, Sorbonne Université UPMC	**Examinatrice**
Philippe Preux, Professeur, Université de Lille	**Examinateur**
Sébastien Verel, Professeur des universités, ULCO	**Directeur de thèse**
Eric Benhamou, PhD, AI for Alpha	**Co-Directeur de thèse**

Model based versus model-free reinforcement learning in asset management

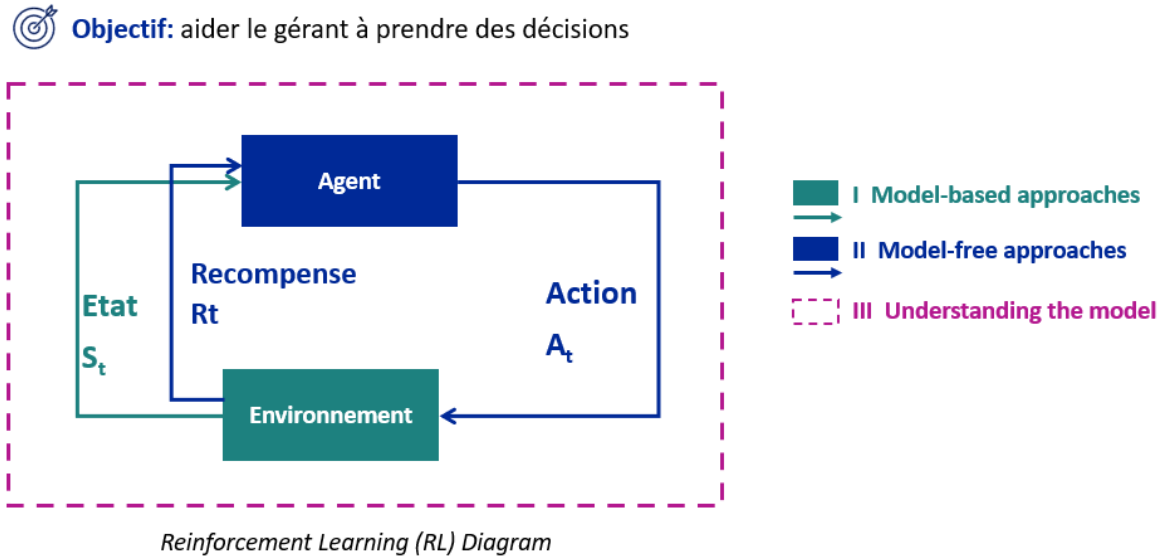
Abstract: The promise of machine learning is to learn rules from raw data without any predefined programming rules. Thus, the machine learns and develops a form of intelligence by identifying these rules by itself, purely relying on data. The limits of this new paradigm are that the computer may loop forever because of an infinite number of rules and that found rules may not be stable over time. This is in particular very relevant in quantitative asset management that aims at finding rules and patterns in financial markets, well known to change behavior over time. In this thesis, we examine the central question of whether machine learning should apply with or without models in the field of quantitative asset management. Rather than supporting one or the other thesis, we examine the two approaches in turn. We initially show that machine learning provides some guidance in selecting model decisions to increase overall performance. We then show that machine learning is able to learn rules directly from data using deep reinforcement learning. We prove that this approach generalizes traditional portfolio optimization methods as it lifts the limits of convex optimization and allows for more informed decisions, by directly linking actions to data and extending the agent's states beyond mean and variance. We examined similarities between supervised and reinforcement learning (RL) and demonstrated that the gradient policy method in RL can be presented as a supervised learning method where the labels are the rewards and the loss function is the cross-entropy function. We conclude the thesis with a Bayesian analysis of the CMAES algorithm and the use of Shapley's value to better understand the machine learning model decision process.

keywords: Machine learning in finance, Deep Reinforcement Learning, Quantitative method

Résumé: La promesse de l'apprentissage automatique est d'apprendre des règles à partir de données brutes sans aucune règle prédéfinie. Ainsi, la machine apprend et développe une forme d'intelligence en identifiant elle-même ces règles, s'appuyant uniquement sur des données. Les limites de ce nouveau paradigme sont que l'ordinateur peut boucler indéfiniment étant donné un nombre infini de règles et que les règles trouvées ne continuent pas dans le temps. Ceci est particulièrement important en gestion d'actifs quantitative qui vise à trouver des règles et des modèles sur les marchés financiers connus pour changer de comportement au fil du temps. Dans cette thèse, nous nous posons la question si l'apprentissage automatique doit s'appliquer avec ou sans modèles en gestion quantitative d'actifs. Plutôt que de soutenir l'une ou l'autre thèse, nous examinons tour à tour les deux approches. Nous montrons dans un premier temps que l'apprentissage automatique permet d'améliorer l'efficacité de modèles en sélectionnant les décisions à retenir. Nous montrons ensuite que l'apprentissage automatique est aussi capable d'apprendre des règles directement à partir des données par apprentissage par renforcement profond. Nous prouvons que cette approche généralise les méthodes traditionnelles d'optimisation de portefeuille, supprimant les limites de l'optimisation convexe et permettant des décisions plus sophistiquées au-delà du cadre moyenne et variance. Nous étudions les similitudes entre apprentissage supervisé et apprentissage par renforcement (RL) et montrons que la méthode de stratégie de gradient en RL s'analyse comme une méthode d'apprentissage supervisé avec des étiquettes données par les récompenses et une fonction de perte spécifiée par l'entropie croisée. Nous concluons la thèse par une analyse Bayésienne de la méthode CMAES et l'utilisation des valeurs de Shapley pour mieux comprendre le processus de décision du modèle d'apprentissage automatique.

Mots clefs: Apprentissage machine en finance, Apprentissage par renforcement profond, méthode quantitative

Resume



Reinforcement Learning (RL) Diagram

La promesse de l'apprentissage automatique est d'apprendre des règles à partir de données brutes sans aucune règle prédéfinie. Ainsi, la machine apprend et développe une forme d'intelligence en identifiant elle-même ces règles, s'appuyant uniquement sur des données. Les limites de ce nouveau paradigme sont que l'ordinateur peut boucler indéfiniment étant donné un nombre infini de règles et que les règles trouvées ne continuent pas dans le temps. Ceci est particulièrement important en gestion d'actifs quantitative qui vise à trouver des règles et des modèles sur les marchés financiers connus pour changer de comportement au fil du temps. Dans cette thèse, nous nous posons la question si l'apprentissage automatique doit s'appliquer avec ou sans modèles en gestion quantitative d'actifs. Plutôt que de soutenir l'une ou l'autre thèse, nous examinons tour à tour les deux approches. Nous montrons dans un premier temps que l'apprentissage automatique permet d'améliorer l'efficacité de modèles en sélectionnant les décisions à retenir. Nous montrons ensuite que l'apprentissage automatique est aussi capable d'apprendre des règles directement à partir des données par apprentissage par renforcement profond. Nous prouvons que cette approche généralise les méthodes traditionnelles d'optimisation de portefeuille, supprimant les limites de l'optimisation convexe et permettant des décisions plus sophistiquées au-delà du cadre moyenne et variance. Nous étudions les similitudes entre apprentissage supervisé et apprentissage par renforcement (RL) et montrons que la méthode de stratégie de gradient en RL s'analyse comme une méthode d'apprentissage supervisé avec des étiquettes données par les récompenses et une fonction de perte spécifiée par l'entropie croisée. Nous concluons la thèse par une analyse Bayésienne de la méthode CMAES et l'utilisation des valeurs de Shapley pour mieux comprendre le processus de décision du modèle d'apprentissage automatique.

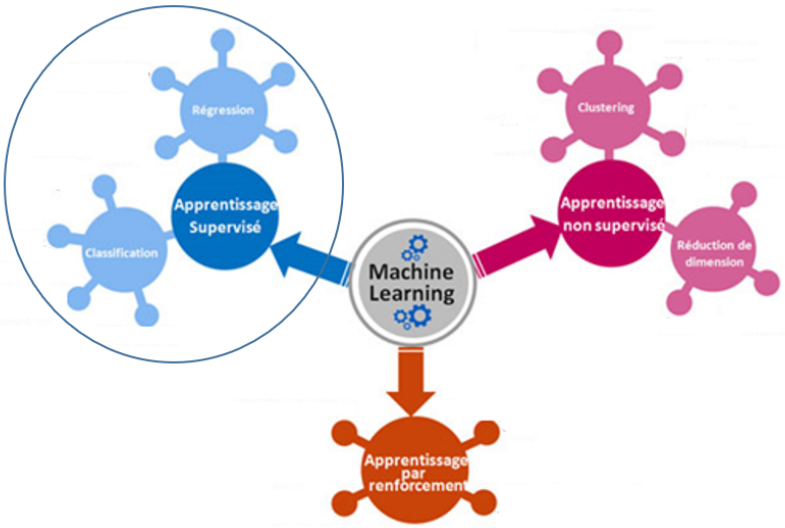


Figure 1: 3 paradigmes du machine learning

Nous nous penchons dans un premier temps sur des méthodes de sélection de variables par apprentissage supervisé.

- La méthode OCA nous permet de sélectionner des variables parmi les variables par bloc et individuelles
- Nous appliquons cette méthode au problème de la prédiction des bons et mauvais trades à partir d'un algorithme de trading de type boîte noire
- La méthodologie tiendra compte de la notion de temporalité

Ces dernières années, les méthodes de pointe pour l'apprentissage supervisé ont de plus en plus exploité les techniques de boosting de gradient, avec des implémentations efficaces telles que xgboost ou lightgbm. L'un des points clés de la création de méthodes efficaces est la sélection des variables (FS). Elle consiste à sélectionner les bonnes variables efficaces et précieuses. Face à des centaines de ces variables, il est essentiel de sélectionner les meilleures. Bien que les méthodes de filtrage et d'enveloppement aient atteint une certaine maturité, les méthodes intégrées sont vraiment nécessaires pour trouver le meilleur ensemble de variables, car ce sont des méthodes hybrides combinant le filtrage et l'enveloppement des variables. Dans ce travail, nous nous attaquons au problème de trouver par l'apprentissage automatique les meilleurs échanges a priori d'une stratégie algorithmique. Nous dérivons cette nouvelle méthode en utilisant l'optimisation par ascension de coordonnées et en utilisant des variables de bloc. Nous comparons notre méthode à la méthode Recursive Feature Elimination (RFE) et à la méthode Binary Coordinate Ascent (BCA). Nous montrons dans un exemple concret la capacité de cette méthode à sélectionner de bonnes transactions a priori. Non seulement cette méthode surpasse la stratégie de trading initiale car elle évite de prendre des trades perdants, mais elle surpasse également les autres méthodes, en ayant à la fois le plus petit ensemble de variables et le plus haut score. L'intérêt de cette méthode

va au-delà de ce simple problème de classification des transactions, car il s'agit d'une méthode très générale permettant de déterminer l'ensemble optimal de variables en utilisant certaines informations sur la relation entre les variables, ainsi qu'en utilisant l'optimisation par ascension de coordonnées.

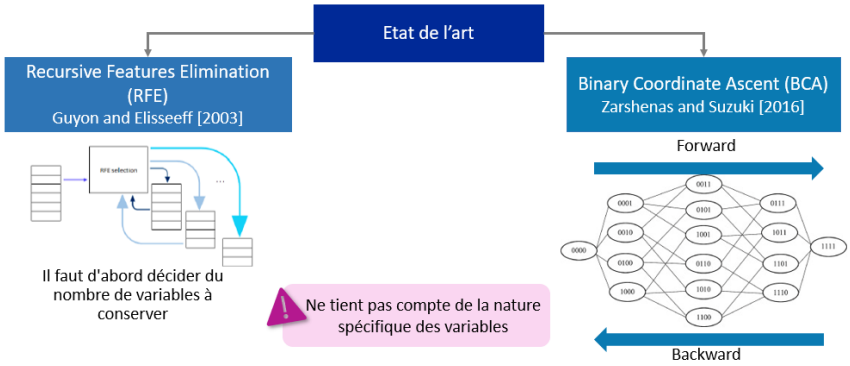
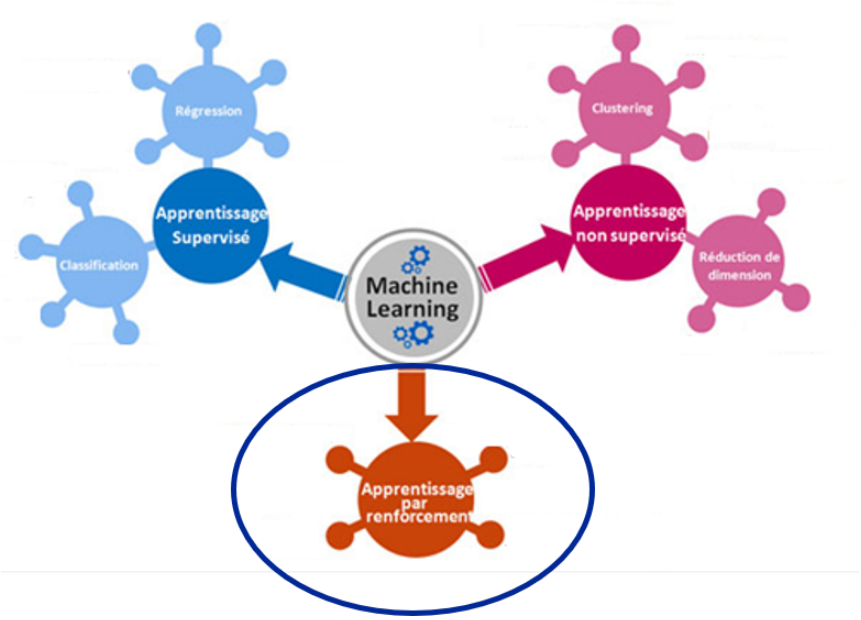


Figure 2: Etat de l'art des méthodes de sélection de variables en apprentissage supervisé

Dans ce chapitre, nous présentons un premier travail de la thèse sur la sélection de trades générés par une méthode de trading algorithmique. Nous montrons que la méthode de trading algorithmique peut être améliorée par des méthodes d'apprentissage supervisé. Le défi est d'utiliser des mesures et des informations regroupées en caractéristiques pour détecter, avant que les ordres ne soient envoyés électroniquement à la bourse, les transactions non réussies hautement probables. Comme la logique de la stratégie de trading algorithmique peut être difficile à comprendre, une méthode d'apprentissage supervisé agnostique peut venir à la rescousse. Cependant, le choix des meilleures variables dans notre ensemble initial de variables est délicat car plus de données fournissent simultanément des informations supplémentaires et du bruit. Nous présentons ici OCA, une nouvelle méthode de sélection des variables qui exploite les similarités entre les variables. Cette méthode n'est pas très exigeante en termes de connaissance des variables et peut sélectionner efficacement les meilleures variables sans tester tous les ensembles de variables possibles. Cela rend la complexité du problème de sélection des variables polynomiale. Lorsqu'elle est mise en œuvre sur des stratégies de cas réels, nous pouvons valider empiriquement que la méthode d'apprentissage supervisé améliore la rentabilité globale du trading. Comme nous demandons à l'algorithme de détecter, dans les opérations de pré-négociation, les candidats hautement infructueux, la méthode est logiquement en mesure de réduire les retraits globaux. La méthode développée ici est assez générale et peut être appliquée à toute classification binaire d'apprentissage supervisé. Dans des travaux ultérieurs, nous aimerions explorer les méthodes d'apprentissage par renforcement afin d'adapter notre méthode aux contraintes de capacité, car il s'agit d'une limitation de l'approche d'apprentissage supervisé. Dans le prochain chapitre, nous étendrons l'approche et utiliserons l'apprentissage automatique pour sélectionner les modèles de ciblage de la volatilité.



Alors que les chercheurs du secteur de la gestion d'actifs se sont principalement concentrés sur des techniques basées sur des techniques financières et de planification des risques telles que la frontière efficiente de Markowitz, la variance minimale, la diversification maximale ou la parité des risques (comme présenté figures 3 et 4) , en parallèle, une autre communauté de l'apprentissage automatique a commencé à travailler sur l'apprentissage par renforcement, et plus particulièrement sur l'apprentissage par renforcement profond, afin de résoudre d'autres problèmes de prise de décision pour des tâches difficiles telles que la conduite autonome, l'apprentissage des robots et, sur un plan plus conceptuel, la résolution de jeux tels que le jeu de Go. Cette partie du manuscrit vise à combler le fossé entre ces deux approches en montrant que les techniques d'apprentissage par renforcement profond (DRL) peuvent apporter un nouvel éclairage sur l'allocation de portefeuille grâce à un cadre d'optimisation plus général qui présente l'allocation de portefeuille comme un problème de contrôle optimal qui n'est pas seulement une optimisation à une étape, mais plutôt une optimisation de contrôle continu avec une récompense différée. Les avantages sont nombreux : (i) DRL fait directement correspondre les conditions du marché aux actions par conception et devrait donc s'adapter à des environnements changeants, (ii) DRL ne repose sur aucune hypothèse traditionnelle de risque financier comme le fait que le risque soit représenté par la variance, (iii) DRL peut incorporer des données supplémentaires et être une méthode à entrées multiples par opposition aux méthodes d'optimisation plus traditionnelles. Nous présentons une expérience avec quelques résultats encourageants en utilisant des réseaux de convolution.

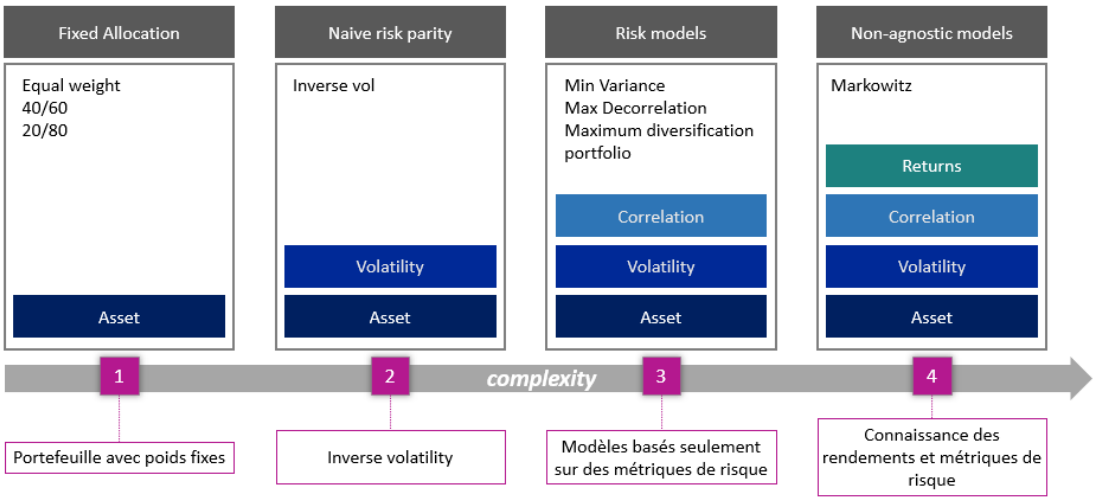


Figure 3: Présentation des méthodes de l'état de l'art pour l'allocation de portefeuille

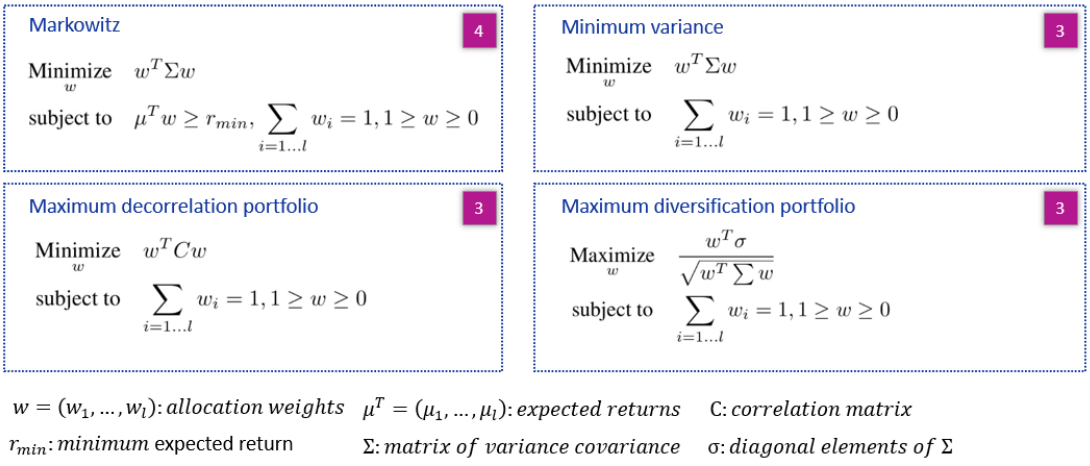


Figure 4: Formalisation mathématique des méthodes d'allocation de portefeuille

Dans ce deuxième chapitre de la partie sans modèle, nous discutons de la façon dont un problème traditionnel d'allocation de portefeuille peut être reformulé comme un problème DRL, en essayant de combler les écarts entre les deux approches. Nous voyons que l'approche DRL nous permet de sélectionner moins de stratégies, améliorant ainsi les résultats globaux par rapport aux méthodes traditionnelles qui sont construites sur le concept de diversification. Nous soulignons également que la méthode DRL peut mieux s'adapter aux conditions changeantes du marché et est capable d'intégrer davantage d'informations pour prendre une décision.

Comme l'illustre l'expérience utilisant une approche DRL, nous obtenons de meilleures performances par rapport aux méthodes traditionnelles. L'architecture de notre algorithme ainsi que l'apport nouveau de variable peut résumer figure 5. Enfin, la prise en compte de la temporalité des données est détaillée figure 6.

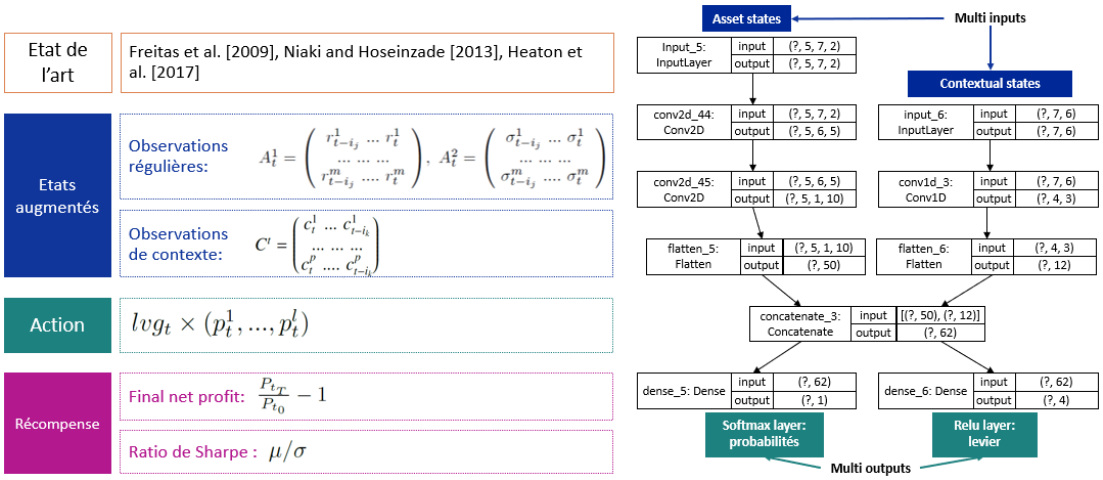


Figure 5: Description du réseau: multi-input du fait de l'apport de données dites de contextes en plus des données de marché. Multi-output du fait de la capacité d'ajouter un effet de levier afin de conforter ou non la confiance qu'a notre modèle en la prédiction des régimes de marché

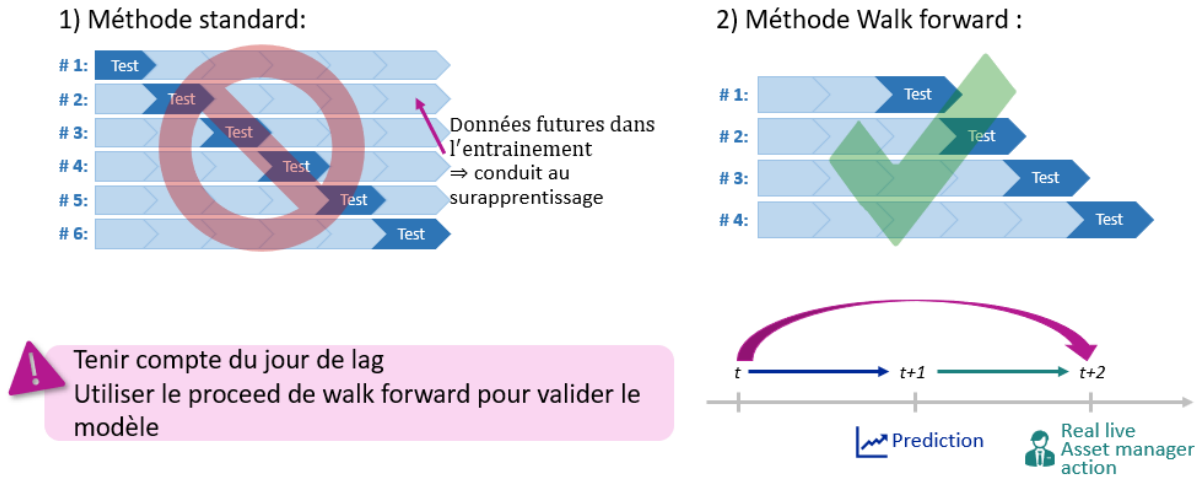


Figure 6: Description des contraintes "métier" des gérants qui doivent tenir compte d'un jour de lag entre la prédiction d'un trade et l'éxecution de celui-ci. De plus, la méthodologie dite de walk forward permet d'assurer la validation du procédé ne prenant aucune donnée future afin de réaliser la prédiction, attestant alors de la robustesse du modèle dans le temps.

Malgré l'intérêt de ce travail, il est possible de l'améliorer car nous n'avons testé que quelques scénarios et un ensemble limité d'hyperparamètres pour nos réseaux convolutifs. Nous devrions effectuer des tests plus intensifs pour confirmer que DRL est capable de mieux s'adapter aux environnements financiers changeants que les méthodes traditionnelles et également étudier l'impact de plus de couches et d'autres choix de conception dans notre réseau.

Compréhension du modèle

Dans la dernière partie du manuscrit, nous présentons deux travaux qui permettent de mieux comprendre les mécanismes de l'apprentissage automatique.

L'une des limites de l'adoption de l'apprentissage automatique en finance quantitative est son aspect boîte noire. Par rapport aux méthodes quantitatives traditionnelles, le ML peut utiliser un grand nombre de paramètres et d'hyperparamètres. En raison du plus grand nombre de paramètres et d'hyperparamètres, le ML sans précaution est plus enclin à l'overfitting que les méthodes quantitatives traditionnelles. En outre, la détermination des hyperparamètres repose sur des méthodes d'optimisation non convexes qui peuvent également sembler mystérieuses.

Dans cette dernière partie, nous examinons d'abord la stratégie d'évolution de l'adaptation de la matrice de covariance (CMA-ES) qui est l'une des méthodes de pointe pour le réglage fin des hyperparamètres en apprentissage automatique. Dans le chapitre 6, nous prouvons que cette approche d'optimisation appelée optimisation boîte noire ou optimisation sans gradient peut être reformulée comme une optimisation bayésienne. Ceci est intéressant car cela conduit naturellement à une nouvelle formulation d'algorithme appelée CMAES bayésienne très similaire à la méthode CMAES initiale. Cela montre également que l'optimisation bayésienne et l'optimisation en boîte noire sont très similaires.

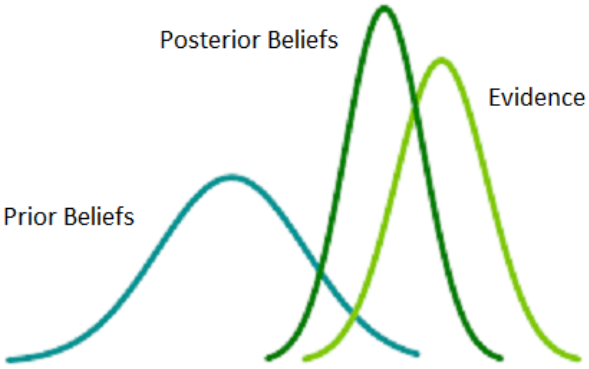


Figure 7: Description de l'update bayésien: au fur et à mesure que nous obtenons de plus en plus d'informations, la loi à postériori devient de plus en plus piqué autour du minimum ou maximum.

En s'appuyant sur la figure 8. Les marchés financiers sont un cas typique d'environnement multi-agents.
Si nous prenons par exemple le SP500 et que nous examinons les interactions, nous pouvons distinguer au moins 3 types d'agents

- les agents du marché qui réagissent très fortement aux nouvelles comme la forte baisse lors de l'annonce du verrouillage de Covid, ou la hausse hebdomadaire lors de l'élection de Trump ou de Biden.
- Les banques centrales, comme le soutien illimité et l'assouplissement quantitatif massif correspondant en mars 2020 après le krach du Covid ou la hausse des taux fin 2018.
- L'action gouvernementale comme la négociation de l'ALENA ou le récent commerce avec la Chine de l'administration Trump (Nafta : Accord de libre-échange nord-américain).

Figure 8: Evolution du SP500 dans le temps

Nous concluons cette partie en présentant quelques méthodes d'explication globale des variables en apprentissage automatique appelées valeurs de Shapley. Dans le chapitre 7, nous illustrons comment les valeurs de Shapley peuvent être utilisées pour obtenir plus d'intuition sur les règles apprises par un modèle dans un modèle LightGBM. En outre, nous pouvons classer les valeurs de Shapley par ordre d'importance, défini comme la moyenne des valeurs absolues de Shapley sur l'ensemble d'apprentissage du modèle pour classer les variables et détecter automatiquement les variables les plus importantes.
La planification des changements de régime sur les marchés financiers est bien connue pour être difficile à expliquer et à interpréter. Un gestionnaire d'actifs peut-il expliquer clairement l'intuition de sa prédiction de changements de régime sur le marché des actions ? Pour répondre à cette question, nous considérons une approche de type gradient boosting decision trees (GBDT) pour planifier les changements de régime sur le SP500 à partir d'un ensemble de 150 variables techniques, fondamentales et macroéconomiques. Nous constatons une amélioration de la précision des GBDT par rapport à d'autres méthodes d'apprentissage automatique (ML) sur les prix à terme du SP500. Nous montrons que la rétention d'un nombre réduit de variables soigneusement sélectionnées apporte des améliorations par rapport à toutes les approches ML. Les valeurs de Shapley ont récemment été introduites de la théorie des jeux dans le domaine de l'apprentissage automatique. Cette approche permet une identification robuste des variables les plus importantes planifiant les crises boursières, et une explication locale de la probabilité de crise à chaque date, par une attribution cohérente des variables.
Nous appliquons cette méthodologie pour analyser en détail la crise financière de mars 2020, pour laquelle le modèle a offert une prédiction hors échantillon opportune. Cette analyse dévoile en particulier le rôle prédictif contrariant du secteur des actions technologiques avant et après le crash.

Comme on peut le voir figure 9, les valeurs moyennes de Shapley représentent essentiellement l'impact moyen sur la sortie du modèle lorsqu'une variables devient "cachée". En outre, l'examen de la corrélation d'une variable avec sa valeur de Shapley donne un aperçu de l'effet de cette variable sur la probabilité d'un krach boursier.

La figure 10 nous permet d'aller un cran plus loin et de comprendre quels ont été les indicateurs haussiers et/ou baissiers conduisant à la probabilité de crash pour un marché concerné. En effet, la figure 10 permet de faire l'analyse suivante:

Indicateur haussier:
Le rendement sur 20 jours du Nasdaq a baissé de 6% ce qui est important, le modèle comprend qu'il y a eu une première baisse et que le marché va alors remonter. Il est intéressant de voir que le modèle utilise des informations sur le Nasdaq pour prédire l'évolution du SP500 (ce qui se comprend bien car 88 des 100 actions du Nasdaq 100 sont dans le SP500). Le modèle est toujours baissier (proba de 61%) mais il comprend que la crise va être rapide et que les marches vont remonter.

Indicateur baissier:
Le taux 3mois US (US libor) sur les marches américains baisse, ce qui résulte du fait que la banque centrale baisse les taux. Or quand cette dernière baisse ses taux, il y a un risque de récession, ce qui est capté par le modèle.

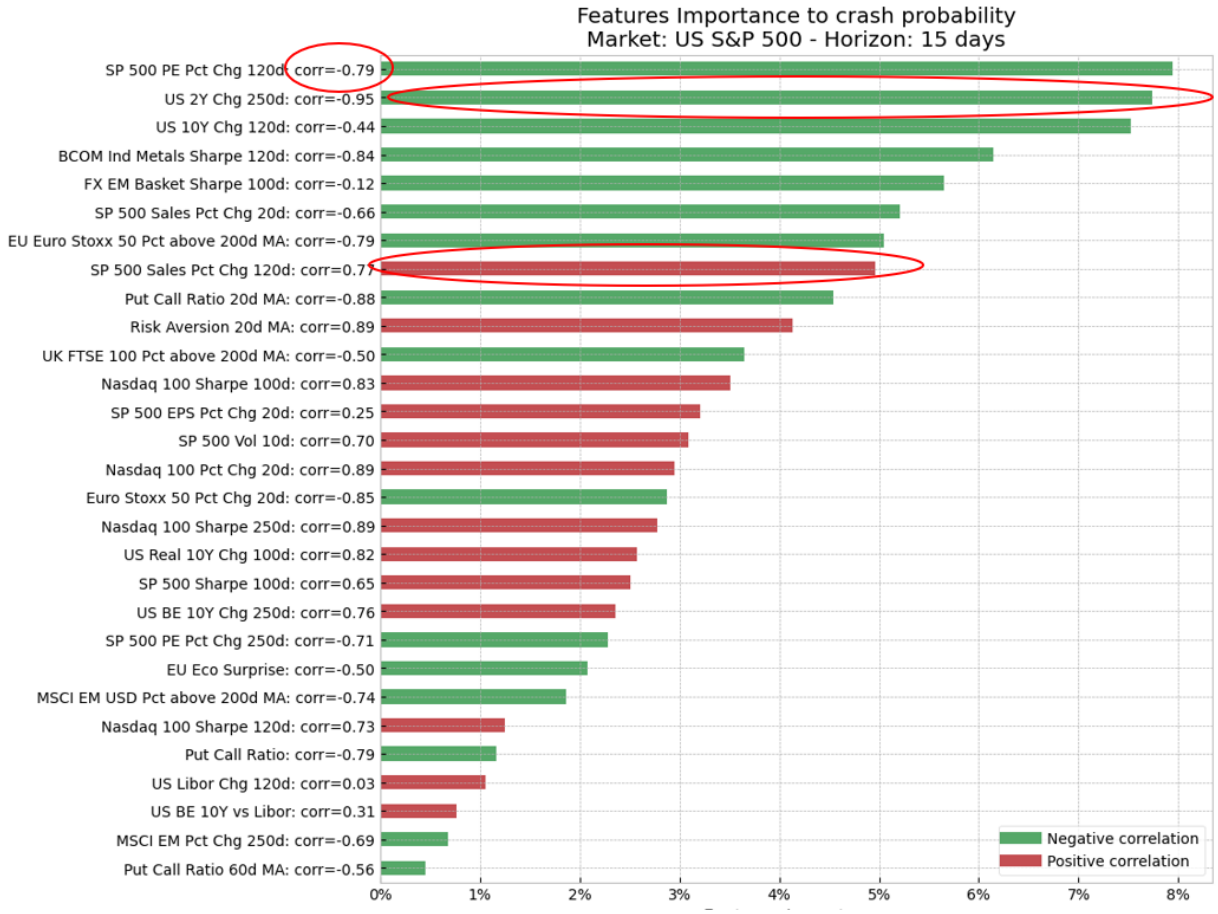


Figure 9: Shapley global

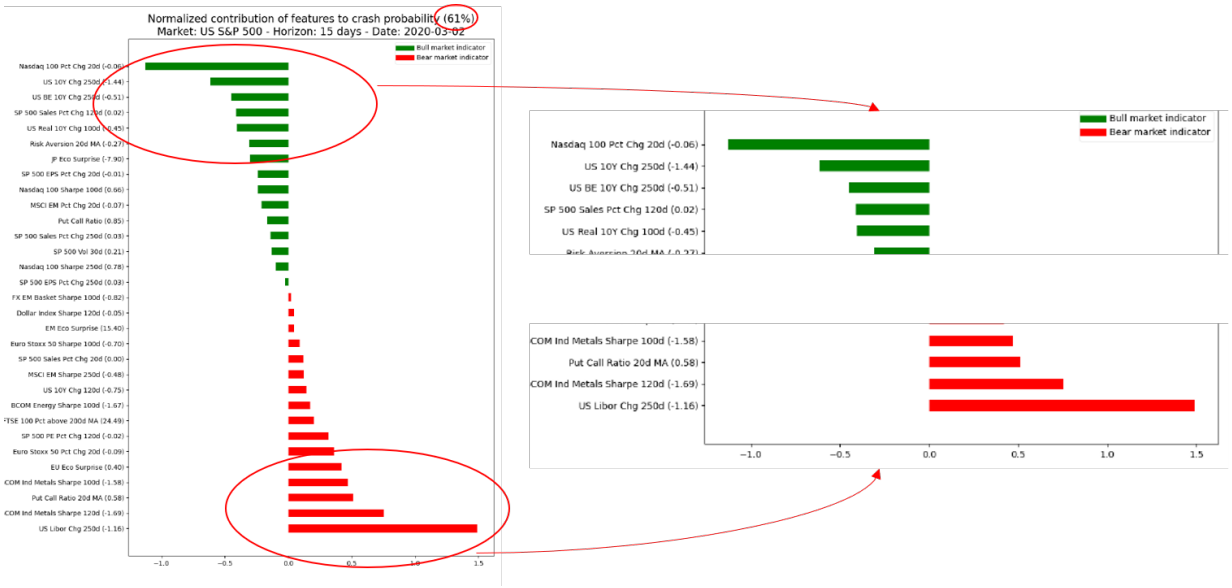


Figure 10: Focus sur la crise du Covid via les valeurs de Shapley.

L'intérêt de l'approche de l'apprentissage automatique est qu'elle peut aider à sélectionner des décisions à partir de modèles quantitatifs traditionnels, créant ainsi ce qu'on appelle un gestionnaire d'actifs augmenté dans une approche basée sur un modèle ou, au contraire, dans une approche sans modèle, aidant à trouver des décisions d'investissement directement à partir d'entrées et de données brutes sans règles préconçues ou biais. L'ensemble des apports de ce manuscrit sont résumés dans la figure 11.

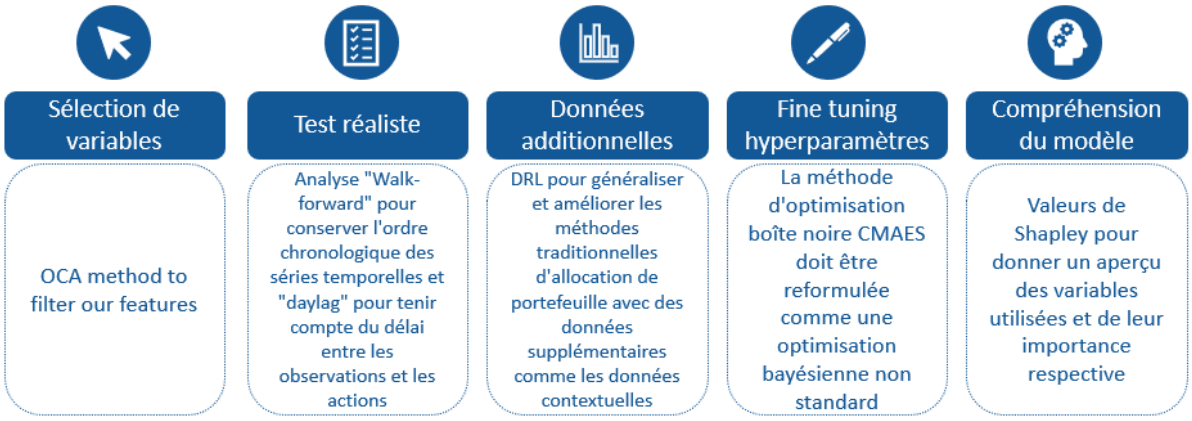


Figure 11: Résumé des méthodologies utilisées au cours du manuscrit de thèse

Acknowledgement

Many people deserve my warm thanks. First, I am immensely grateful to my two supervisors, Sébastien Verel, my academic supervisor, Professeur des Universités at Laboratoire d'Informatique, Signal et Image de la Côte d'Opale and Eric Benhamou, my CIFRE supervisor within Ai Square Connect and lately Ai for Alpha, Head of Research at AI for Alpha. Both of them have been inspiring in terms of research and mentoring.

In particular, I am very indebted to Eric Benhamou for guiding me within the machine learning research world. The area of research in machine learning is growing at an unseen pace. Eric is among the few researchers that can still have encyclopedic knowledge in the field. I am also grateful for his dedication and the long hours spent with me. I am also grateful to Sébatien Vérel who has been of great help in orientating my research and encouraging me during all this time.

I would like to thank the AI Square Connect and lately the Ai for Alpha team for its complete support for this thesis, in particular Beatrice Guez, Jean Jacques Ohana, Stephane Fadda, and Nicolas Paris for fruitful conversations and interactions. Among these, Beatrice Guez has been very supportive of this thesis and always keen to give me some spare time to work on prospective research works. Jean Jacques Ohana has been a great co-author and a powerful companion in many research works. He did not hesitate to transmit his passion for hard work, his knowledge, and his rigor.

Another mention and thank go to my other coauthors: Sandrine Ungari, Abhishek Mukhopadhyay, Serge Tabachnick, Sui Kai Wong, François Chareyron, and Corentin Bourdeix.

I would like to thank my reviewers Jonathan Weber and Emmanuel Lepinette for their meaningful remarks. A special thank to my jury members Laetitia Jourdan, Nicole El Karoui and Philippe Preux for their involvement and pertinent reflections during the defense.

I am very grateful to my family and friends (the Tahans) for their unconditional support and patience in difficult times. A special thought for my parents Karine and Olivier, and my two brothers Mickaël and Jordan. I don't forget where I come from, I am the man I am thanks to their unfailing love and support. A tender thought comes to my grandmother Jeanne who, I know, followed the end of my thesis from above. A huge thank to my grandparents Armand (Papi) and Sonia (Nini) who have supported me since the first day of my thesis and who, I hope, will be proud to see their grandchild grow up. Last but not least, I would like to thank my best friend, wife and support Yaël, without whom this PhD thesis would not have been possible. It has been challenging to finish writing this Ph.D. dissertation while organizing my wedding and I thank my wife for having been so comprehensive and supportive. The evening rehearsals and reviews of the slides will remain a memory forever.

Contents

Introduction

In this thesis, done in partnership between academia and a startup, we study the problem of using machine learning to take informed decisions in quantitative asset management. By quantitative asset management, we refer to a systematic approach to investment service, using some financial times series and some data to make profitable investment decisions for a financial investor.

Taking decisions that generate financial benefits for an investor is a very challenging problem as financial markets are well known to be unpredictable and to change behavior. Moreover, when investment rules are disclosed to a large population, they often stop functioning because of overcrowding and potential bubbles. Being able to adapt and modify systematic strategies to reflect structural changes in financial markets is therefore an area of active research. In this thesis, we study whether machine learning can help solve this difficult question thanks to its promise to continuously learn and adapt. However, instead of trying to address a very large problem, we restrict ourselves to a smaller question on the interest of combining machine learning with traditional quantitative methods. In the world of machine learning and in particular, reinforcement learning the approach of using a model to represent the dynamic of the financial market is called a model-based approach. In contrast, using no model is called a model-free approach. Compared to supervised learning which aims at finding a mapping between inputs and outputs thanks to the guidance of a loss function, reinforcement learning tackles the problem of learning a mapping function between states and actions in order to maximize a reward. In our setting, the states are financial variables or features that represent the financial markets status and the actions are the allocations or weights on various assets. If we use a model, we already have some allocations recommended by the model and the machine learning task is to take or not the recommendations of the quantitative model. This extra layer of the model can help correct the inaccuracy of the initial model by ignoring what are thought to be wrong decisions. In contrast, using a model-free approach implies having no initial decision and letting the model decide freely the final allocation.

Context

Artificial intelligence in finance sits at the intersection of a number of established disciplines, including financial econometrics, financial mathematics, statistical calculation, probabilistic programming, dynamic programming, quantitative finance, and pattern recognition. With the trend towards growing computing resources combined with easier access to data and larger datasets, applying artificial intelligence or more precisely machine learning has become a central area of focus and scrutiny for quants, researchers, and more generally any data scientist interested in applying machine learning to finance.

However, despite this growing interest illustrated by the rising number of publications associating machine learning and finance, much of the field remains a mystery, outside of engineering-based research groups and business activities. Huang et al. [2020] for instance in its literature review of deep learning in finance and banking emphasizes the growth rate of publication of machine learning in finance and its broad spectrum with a strong emphasis in NLP and usage for commercial banking. Mazrouei and Nobanee [2020] in its mini-review of machine learning draws the same conclusion that machine learning publications are growing at a very high rate and are predominant in usage for commercial banking. Dixon et al. [2020]is a nice tentative to overcome the gap in machine learning for quantitative finance. It presents a unified treatment of ML and various statistical and computational disciplines for quantitative finance, with in particular connections to financial econometrics and discrete time stochastic control. Goodell et al. [2021] is a nice tentative to identify foundations, themes, and research clusters from bibliographical analysis in finance and illustrate that most of the work of machine learning in quantitative finance remains unpublished.

This is because machine learning in finance is challenging due to the constant adaptive nature of financial markets. Indeed, the promise of machine learning to learn rules from raw data is quite daunting in financial markets due to the intricacy between predictive and noisy data as well as their changing nature. By changing nature, we mean that data that are predictive during certain periods may not be predictive in the future. Equally, data that are noisy in the first place, may become predictive given that rules change and adapt constantly, making the machine learning exercise highly difficult and challenging. For financial professionals, accustomed to the regulatory saying *past performance is not indicative of future results*, this comes as no surprise. Indeed, this naturally leads to the essential problem examined in this thesis: Should we do model-based or model-free machine learning? By the former, we mean the more pragmatic approach of using machine learning to select quantitative models that are based on human intuition and have been validated by years of practice and experience, and just need to be slightly adapted to changing regimes. By the latter, we mean the idealized approach of throwing data to the machine and letting the machine find the rules.

Contributions and layout

Because the open question is so vast, we will restrict ourselves to asset management in this thesis. Thus, we examine in this manuscript the central question of whether machine learning should apply without or with models in the field of quantitative asset management. Rather than supporting one or the other thesis, we examine the two approaches in turn.

We will start our journey with part I where we take the point of view of a complementary machine learning approach. It intuitively augments the asset manager by providing him decision tools thanks to machine learning to decide among various established models. In this part, we use machine learning to select various models that are obtained through traditional approaches. The only knowledge from these models are a very general overview of their methods and the results that they generate. With this setting, the approach is very general as we only need to use the outputs of these quantitative models without having access to their internal logic and code.

Chapter 1 consists in choosing in advance, financial transactions described by a model to avoid or at least try to avoid future losses. This is done thanks to supervised learning and in particular, gradient boosting decision trees (GBDT) techniques, which are well documented on various machine learning platforms to provide state-of-the-art methods for classification given small data sets as opposed to deep learning that are data intensive methods and struggle on small data sets. Typical implementations of GBDT are XGBoost as presented in Chen and Guestrin [2016]. XGBoost which stands for (eXtreme Gradient Boosting) is in fact an open-source software library which provides a regularizing gradient boosting framework for multiple languages like C++, Java, Python,R,Julia, Perl, and Scala at least. It is cross platform and works on Linux, Windows, and macOS. LightGBM as presented Ke et al. [2017] is a more recent implementation of GBDT that has been originally developed by Microsoft. Compared to XGBoost, its construction of trees is not done at a tree level-wise — row by row — as XGBoost but rather at the leaf-wise level. The choice to use a growth at a leaf-wise level is motivated by the experimental remark that this growth at leaf-level is believed to yield the largest decrease in loss. Last but not least, another implementation of XGBoost has been proposed by Prokhorenkova et al. [2018] and is called Catboost.

The key challenge in GBDT lies in Feature Selection (FS). A lot of methods have been presented. Tuv et al. [2009] using tree-based ensembles and redundancy elimination presents a method to generate a compact subset of non-redundant features. Indeed Guyon et al. [2005] through a detailed analysis of the NIPS 2003 feature selection challenge emphasized the importance of feature selection. Likewise, Mangal and Holm [2018] illustrates the importance of features selection in a classification problem of stress hot spot and shows experimentally that features selection is one of the most important parts in the conception of the machine learning model. Ghalwash et al. [2016] shows that coordinate descent optimization can provide an efficient algorithm to select structured features.

More generally, feature selection is the process of selecting the variables to provide to the model to do its learning step. When facing hundreds of these features, it becomes critical to select the best features. While filter and wrapper methods have come to some maturity, embedded methods are truly necessary to find the best features set as they are hybrid methods combining features filtering and wrapping. In our work, we tackle the problem of finding through machine learning the best a priori trades from an algorithmic strategy. We derive this new method using coordinate ascent optimization and using block variables. We compare our method to Recursive Feature Elimination (RFE as presented in Mangal and Holm [2018]) and Binary Coordinate Ascent (BCA as presented in Zarshenas and Suzuki [2016]). We show in a real-life example the capacity of this method to select good trades a priori. Not only does this method outperforms the initial trading strategy as it avoids taking losing trades, but it also surpasses other methods, having the smallest feature set and the highest score at the same time. The interest of this method goes beyond this simple trade classification problem as it is a very general method to determine the optimal feature set using some information about features relationship as well as using coordinate ascent optimization.

Chapter 2 focuses on the choice of targeting models to favor thanks to deep reinforcement learning. In the context of risk-based portfolio construction and proactive risk management, finding robust predictors of future realized volatility is paramount to achieving optimal performance. Volatility has been documented in economics literature to exhibit pronounced persistence with clusters of high or low volatility regimes and to mean-revert to a normal level as presented Hocquard et al. [2013]. Perchet et al. [2016] shows that controlling volatility through strategies of volatility targeting can lead to successful strategies. Moreover, with managed strategies, tail risk meaning extreme events can be mitigated Dreyer and Hubrich [2017]. This comes at no suprise that Generalized Autoregressive Heteroskedastic (GARCH) models as presented in the prize-winning work of Bollerslev [1986] can effectively help in predicting more accurately volatility. From a Reinforcement Learning (RL) point of view, this process can be interpreted as a model-based RL approach (like in Deisenroth and Rasmussen [2011b]), where the goal of the models is twofold: first, to represent the volatility dynamics and forecast its term structure and second, to compute a resulting allocation to match a given target volatility: hence the name "volatility targeting method for risk-based portfolios". However, the resulting volatility model-based RL approaches are hard to distinguish as each model results in similar performance without a clear dominant one. We, therefore, present an innovative approach with an additional supervised learning step to predict the best model(s), based on the historical performance ordering of RL models. Our contribution shows that adding a supervised learning overlay to decide which model(s) to use provides improvement over a naive benchmark consisting in averaging all RL models. A salient ingredient in this supervised learning task is to adaptively select features based on their significance, thanks to minimum importance filtering.

In part II, we make a clean sweep of any model and try to learn rules directly from data.

We open this part by presenting in chapter 3 an application of deep reinforcement learning to detect pattern crisis and consequently dis-investing. Indeed, Deep Reinforcement Learning (DRL) has reached superhuman levels in complex tasks like game solving (as for the game of Go Silver et al. [2017], or for the game of StarCraft II Vinyals et al. [2019]), and autonomous driving Wang et al. [2018]. However, it remains an open question whether DRL can reach human level in applications to financial problems and in particular in detecting pattern crisis and consequently dis-investing. In this chapter, we present an innovative DRL framework consisting in two sub-networks fed respectively with portfolio strategies past performances and standard deviations as well as additional contextual features. The second sub network plays an important role as it captures dependencies with common financial indicators features like risk aversion, economic surprise index and correlations between assets that allows taking into account context-based information. We compare different network architectures either using layers of convolutions to reduce network's complexity or LSTM block to capture time dependency and whether previous allocations are important in the modeling. We also use adversarial training to make the final model more robust. Results on the test set show this approach substantially over-performs traditional portfolio optimization methods like Markovitz and is able to detect and anticipate crisis like the current Covid one.

Chapter 4 goes beyond and shows that the deep reinforcement learning (DRL) approach applied to portfolio allocation makes it possible to generalize the question by posing it as an optimal control problem which is no longer just an optimization in one step, but rather a continuous optimization control with a delayed reward. It starts by a comparison between traditional portfolio allocation

methods like Markowitz efficient frontier (Markowitz [1952b]), minimum variance (Chopra and Ziemba [1993], Kritzman [2014]), maximum diversification or equal risk parity (Roncalli and Weisang [2016]) and deep reinforcement learning methods. This chapter aims to bridge the gap between these two approaches by showing Deep Reinforcement Learning (DRL) techniques can shed new lights on portfolio allocation thanks to a more general optimization setting that casts portfolio allocation as an optimal control problem that is not just a one-step optimization, but rather a continuous control optimization with a delayed reward. The advantages are numerous: (i) DRL maps directly market conditions to actions by design and hence should adapt to changing environment, (ii) DRL does not rely on any traditional financial risk assumptions like that risk is represented by variance, (iii) DRL can incorporate additional data and be a multi-inputs method as opposed to more traditional optimization methods. We present on an experiment some encouraging results using convolution networks.

We conclude this part with a more theoretical point of view. In chapter 5, we explore the similarities between supervised and reinforcement learning. Reinforcement learning (RL) is about sequential decision making and is traditionally opposed to supervised learning (SL) and unsupervised learning (USL) Hastie et al. [2009b]. In RL, given the current state, the agent makes a decision that may influence the next state as opposed to SL (and USL) where the next state remains the same, regardless of the decisions taken, either in batch or online learning Sutton and Barto [2018]. Although this difference is fundamental between SL and RL, there are connections that have been overlooked Bishop [2007]. In particular, we prove in this chapter that the gradient policy method can be cast as a supervised learning problem where true labels are replaced with discounted rewards. We provide a new proof of policy gradient methods (PGM) that emphasizes the tight link with the cross entropy and supervised learning.

We end the thesis with part III to gain intuition and to better explain what machine learning models are doing. In this final part, we provide more intuition about black box optimization that is widely used for hyper parameter tuning. We also investigate the use of Shapley's value to better understand the marginal contribution of variables.

In chapter 6, we demonstrate that black box optimization methods widely used in hyper parameters determination can be interpreted as Bayesian optimization shedding new light on the correspondences between these two traditionally opposed types of optimizations. Indeed, this novel theoretically sound approach for the celebrated CMA-ES algorithm (as introduced in Hansen and Ostermeier [2001] and later developed in numerous papers as presented in Akimoto et al. [2016] or through information geometry optimization Ollivier et al. [2017]) shows that CMAES is quite similar to assuming a conjugate prior distribution for the the parameters of the multivariate normal distribution for the minimum follow and deriving the optimal update at each iteration step. Not only provides this Bayesian framework a justification for the update of the CMA-ES algorithm but it also gives two new versions of CMA-ES either assuming normal-Wishart or normal-Inverse Wishart priors, depending whether we parametrize the likelihood by its covariance or precision matrix. We support our theoretical findings by numerical experiments that show fast convergence of these modified versions of CMA-ES.

We end with chapter 7 which discusses the use of Shapley's value. Shapley values, originally presented in game theory by Noble prize Loy Shapley Shapley [1953] and adapted to machine learning in Lundberg and Lee [2017], make it possible to better understand the impact and the

marginal contribution of a variable among all the variables and thus provide an explanatory tool on the impact of this variable. Using Shapley values, we are able to better understand the significance and influence of all our financial variables in our model. In particular, we can understand if a variable or features presents a mean reverting or trend behavior.

Finally, in conclusion, we summarize the major ideas developed in this thesis and present some future ideas and new research directions.

All in all, this thesis has led to 12 publications listed in the appendix section. It is worth also mentioning that this thesis has been done in a CIFRE framework, leading to cover various materials and questions largely driven by business logic and constraints. We voluntarily restricted ourselves to a subset of the various works done during the thesis to keep things coherent.

PART I

Model-based approaches

Overview

Machine learning (ML) gives computers the ability to learn without being explicitly programmed (see Samuel [1959]). It makes computers learn and improve automatically with experience. This experience is crucial in the learning process and motivates some initial framework or model-based approach that processes the data. The objective of the Machine learning part is to ameliorate the initial financial model.

In this part, we will consider an initial financial time series denoted by $X_1, ..., X_n$ where $X_t, t \in \mathbb{N}$ is a chronological time series of decisions obtained by a financial model. The objective of Machine Learning is to improve the result obtained by this financial model by selecting when to follow the decision initiated by this initial financial time series. This model can be either a trading strategy as presented in chapter 1 or a quantitative model predicting volatility in financial markets in the context of portfolio allocation according to volatility target methods as presented in chapter 2.

Following the presentation in Mitchell et al. [1990], this problem can be examined initially as a Supervised learning task, where we are learning a function from example data made of pairs of input and correct output. We will say that we should follow the decision $X_t, t \in \mathbb{N}$ if its outcome according to some financial criterion is positive. Hence we will be able to associate to each decision a binary label stating whether it is a good or bad decision. This is the subject of chapter 1.

It can also be examined as a Reinforcement learning task, where learning is done with no knowledge of an exact output for a given input but rather some indication that the decision taken is better or worse according to some complex reward function. In contrast to the first approach that only focuses on the immediate impact of the decision, this approach through the reward function allows us to measure the accumulated impact of the various decisions that are not independent and influence each other. This is the subject of chapter 2.

The terminology of model-based is mostly used in Reinforcement Learning. Indeed, machines learn differently than humans. For example, you probably didn't learn the difference between a positive and a negative movie review by analyzing tens of thousands of labeled examples of each. However, there are certain sub-fields in machine learning that are very similar to how we learn.

Reinforcement learning (RL) is a field that has been around for decades. More recently, it has gained momentum thanks to the integration of deep neural networks (deep reinforcement learning) and the resulting cumulative newsworthy success. However, the heart of RL is concerned with how to make decisions in a given environment and take courses of action to maximize rewards. Or, to put it in a more personal interpretation, what steps do you need to improve your health and fitness, or to get promoted at work, or to make a good investment decision? We tend to figure out the optimal approach to achieving such goals through some degree of trial and error, evolving their strategies based on feedback from the environment.

At a basic level, RL works pretty much the same. Of course, backed by computing power, various strategies ("policies" in the RL literature) can be explored much faster than what a human can do, often with very impressive results (especially in the case of simple environments).On the other hand, lacking the prior knowledge that humans bring to new situations and environments, RL approaches tend to need to explore many more policies than a human would do before finding the best one.

This is precisely what model-based reinforcement learning attempts to overcome. More precisely,

model-based RL attempts to overcome the issue of a lack of prior knowledge by enabling the agent — whether this agent happens to be a robot in the real world, a bot in the virtual world, or just some software that takes actions — to construct a functional representation of its environment.

In quantitative finance where the potential choice of decisions, data, and solutions is enormous, model-based reinforcement learning may certainly play an important role in better understanding financial markets. In a sense, understanding means forming a predictive model of financial markets and using it to get to make an informed decision. In contrast to the model-free approach that will be presented in the second part of this thesis, model-based RL has many advantages. First, it tends to be more sample efficient than model-free RL. This means less data is required to learn the policy, which is very relevant to non-high frequency quantitative finance that uses only daily data. Saying differently, model-based reinforcement learning leverages the information it has learned about its environment so that it can not only react but also plan, forecast, and simulate a sequence of actions without having to perform them directly in the real environment.

A related benefit is that model-based RL can potentially be repurposed for other goals and tasks, thanks to the modeling process. Learning a single policy is good for a single task, but if we can predict the dynamics of our environment, we can generalize those insights to multiple tasks. We can factor in the model-based approach not only an interest in the final net performance of a financial strategy but also some risk-reward trade-off. Finally, having a model means that you can determine some degree of model uncertainty, so you can determine how confident you should be about the resulting decision-making process.

Moving on to the cons of model-based RL (or pros of model-free RL), one of the biggest is the need to learn the policy (the overall strategy to maximize the reward), not just the model. It exacerbates the degree of potential error. In other words, there are two different sources of approximation error in model-based reinforcement learning, but only one in model-free reinforcement learning. For similar reasons, model-based approaches tend to include, by definition, extra layers of modelization and hence computation steps making it potentially more computationally intensive and demanding.

It's worth noting that we do not need to take a binary decision of using a model-based or model-free approach. Some of the most effective recent approaches have combined model-based and model-free strategies. Perhaps this is not so surprising given the evidence that the human brain employs both model-free and model-based decision-making strategies in parallel, with each dominating in different circumstances. Combining the two approaches will however be beyond the scope of this thesis and left for future investigation.

1 Trade Selection with Supervised Learning and Optimal Coordinate Ascent

1.1 Motivations

In recent years, state-of-the-art methods for supervised learning have increasingly exploited gradient boosting techniques, with mainstream efficient implementations such as xgboost or lightgbm. One of the key points in generating proficient methods is Feature Selection (FS). It consists in selecting the right valuable effective features. When facing hundreds of these features, it becomes critical to select the best features. While filter and wrapper methods have come to some maturity, embedded methods are truly necessary to find the best features set as they are hybrid methods combining features filtering and wrapping. In this work, we tackle the problem of finding through machine learning the best a priori trades from an algorithmic strategy. We derive this new method using coordinate ascent optimization and using block variables. We compare our method to Recursive Feature Elimination (RFE) and Binary Coordinate Ascent (BCA). We show in a real-life example the capacity of this method to select good trades a priori. Not only does this method outperforms the initial trading strategy as it avoids taking losing trades, but it also surpasses other methods, having the smallest feature set and the highest score at the same time. The interest of this method goes beyond this simple trade classification problem as it is a very general method to determine the optimal feature set using some information about features relationship as well as using coordinate ascent optimization.

1.2 Introduction: a motivating example

In financial markets, algorithmic trading has become more and more standard over the last few years. The rise of the machine has been particularly significant in liquid and electronic markets such as foreign exchange and futures markets reaching between 60 to 80 percent of total traded volume (see for instance Chan [2013], Goldstein et al. [2014] or Chaboud et al. [2015] for more details on the various markets). These strategies are even more concentrated whenever there are very fast market moves as reported in Kirilenko et al. [2017]. These algorithmic trading strategies typically rely on historical statistics. The main concept is to find some trading signals

and information that identifies a pattern or trend with a high probability of repetition. As desirable as it may be, the perfect algorithm is the one with the highest accuracy in terms of identifying the targeted pattern and with the smallest number of losing trades.

If we want to increase robustness and bring additional firewalls to the trading strategy, it makes sense to add supplementary logic with the use of a supervised learning method. The question is to empirically validate whether a supervised machine learning method can a priori identify bad or good trade and hence select among the systematic trades spawned by our algorithmic trading strategy. This is a typical supervised learning classification problem, very similar to the boilerplate example of identifying spam in emails. The complexity of this challenge is to identify features that are relevant to assist the machine in being able to in advance determine the chance of success of a machine-based trade.

This motivates an efficient method to select among a large set of features the ones that create an efficient algorithm. This is precisely the subject of this chapter. It is organized as follows. We first present the supervised learning classification problem. We then present the Optimal Coordinate Ascent (OCA) algorithm that enables us to select the Pareto optimal features set. The key contribution of this method is to exploit similarities between features and hence reduce the optimization search within categories as well as use coordinate ascent to a polynomial complexity problem as we modify the initial problem whose complexity is NP-hard to a new problem whose complexity is just polynomial. We then present results on a real-life trading policy. We show that there is a substantial improvement compared to the original strategy. We conclude with further work.

1.3 Experience description

1.3.1 Challenge description

A trading strategy is usually defined with some signal that generates a trading entry. But once we are in position, the next question is the trading exit strategy. There are multiple methods to handle efficient exits, ranging from fixed target and stop loss, to dynamic target and stop loss. Indeed, to enforce success and crystallize gain or limit loss, a common practice is to associate the strategy with a profit target and stop loss as described in various papers (Labadie and Lehalle [2010], Giuseppe Di Graziano [2014], Fung [2017], or Vezeris et al. [2018]). The profit target ensures that the strategy locks in real money the profit realized and is materialized by a limit order. The stop loss that is physically generated by a stop order safeguards the overall risk by limiting losses whenever the market backfires and contradicts the presumed pattern. To keep things simple we will hereby examine a trading strategy that has a fixed profit target and stop loss. It generates about 1500 trades over a period of 10 years. For each of these trades, we make some measurements to get 135 features. The challenge is from these features to predict which trade is going to be successful. If we give brutally these features to a gradient boosting method like xgboost or lightgbm, the algorithm performs poorly as it is swamped by too much noisy data. The features that are provided are proprietary indicators whose identity and source are ignored by our machine learning algorithm. The challenge here is to find the optimal features set for our gradient boosting method. The learning process is summarized by figure 1.1.

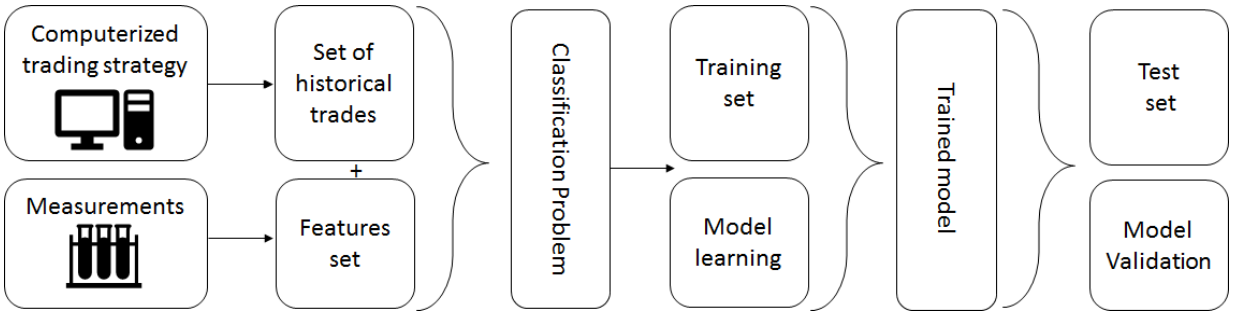


Figure 1.1: Learning process for our trade selection challenge. We first use a proprietary trading strategy that generates some sample trades. We take various measures before the trades are executed to create a feature set. We combined these to create a supervised learning classification problem. Using xgboost method and OCA, we learn model parameters on a train set. We monitor the overall performance of the trading strategy on a separate test set to validate scarce overfitting.

1.3.2 Feature selection

Feature selection is also known as variable or attribute selection. It is the selection of a subset of relevant attributes in our data that are most relevant to our predictive modeling problem. It has been an active and fruitful field of research and development for decades in statistical learning. It has proven to be effective and useful in both theory and practice for many reasons: enhanced learning efficiency and increasing predictive accuracy (see Mitra et al. [2002]), model simplification to ease its interpretation and improve performance (see Almuallim and Dietterich [1994], Koller and Sahami [1996] and Blum and Langley [1997]), shorter training time (see Mitra et al. [2002]), curse of dimensionality avoidance, enhanced generalization with reduced overfitting, implied variance reduction. Both Hastie et al. [2009a] and Guyon and Elisseeff [2003] are nice references to get an overview of various methods to tackle features selections. The approaches followed vary. Briefly speaking, the methods can be sorted into three main categories: Filter methods, Wrapper methods, and Embedded methods.

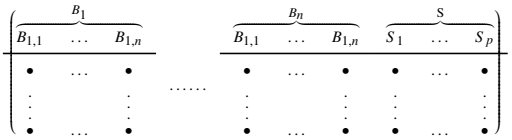
However, these methods do not exploit some particularities of our features set. We are able to regroup features among families. We call these features block variables. A typical example is to regroup variables that are observations of some physical quantity but at a different time (like the speed of the wind measure at different hours for some energy prediction problem, like the price of a stock in an algorithmic trading strategy for financial markets, like the temperature or heartbeat of a patient at different time, etc ...).

1.4 OCA Method

The approach adopted here is the method referred to as the Optimal Coordinate Ascent (OCA) method that is described in Saltiel and Benhamou [2018]. Formally, we can regroup our variables into two sets:

- The first set encompasses $B_1 \dots B_n$. These are called block variables of different lengths L_i. Mathematically, the Block variables are denoted by B_i with B_i taking value in $\mathbb{R}^{L_i}$, $\forall i \in 1 \dots n$
- The second set is denoted S and is a block of p single variables.

Graphically, our variables look like that:



In addition, we have N variables split between block variables and single variables, hence $N = N_B + p$ with $N_B = \sum_{i=1}^{n} L_i$.

1.4.1 Algorithm description

Our algorithm works as follows. We first fit our classification model to find a ranking of features importance. The performance is computed with the Gini index for each variable. We then keep the first k best ranked features for each block $B_1 \dots B_n$ in order to find the best initial guess for our coordinate ascent algorithm. Notice that the set of unique variables is not modified during the first step of the procedure. The objective function is the number of correctly classified samples at each iteration. We then enter the main loop of the algorithm. Starting with the vector of $\left(k, \dots, k, \ \mathbb{1}_p^T\right)$ as the initial guess for our algorithm, we perform our coordinate ascent optimization in order to find the set with the optimal score and the minimum number of features. The coordinate ascent loop stops whenever we either reach the maximum number of iterations or the current optimal solution has not moved between two steps.

We summarize the algorithm in the pseudo code 1. We denote by ε the tolerance for the convergence stopping condition. To control early stop, we use a precision variable denoted by $\varepsilon_1, \varepsilon_2$ and two iteration maximum Iteration $\max_1$ and Iteration $\max_2$ that are initialized before starting the algorithm. We also denote Score$(k_1, \ \dots, \ k_n, \ \mathbb{1}_p)$ to be the accuracy score of our classifier with each B_i block of variables retaining k_i best variables and with single variable all retained.

Remark 1.1. The originality of this coordinate ascent optimization is to regroup variables by block, hence it reduces the number of iterations compared to Binary Coordinate Ascent (BCA) as presented in Zarshenas and Suzuki [2016] The stopping condition can be changed to accommodate other stopping conditions.

Remark 1.2. The specificity of our method is to keep the j best representative features for each feature class, as opposed to other methods that only select one representative feature from each group, ignoring the strong similarities between each feature of a given variable block. This takes in particular the opposite view of feature Selection with Ensembles, Artificial Variables, and Redundancy Elimination as developed in Tuv et al. [2009].

1.5 Convergence analysis on stylized objective function

Although it may be hard to determine the convergence speed for a real-life example, under some weak conditions on the function f of our optimization problem, we can prove that the convergence speed is linear. To be more specific, let us assume we examine the following optimization program: $\min_x f(x)$. f is called the objective function. We denote by e_i the traditional vector with 0 for any coordinate except 1 for coordinate i. It is the vector of the canonical basis.

Algorithm 1 OCA algorithm

J Best optimization
We retrieve features importance from a fitted model
We find the index $k^\star$ that gives the best score for variables block of the same size k:
$k^\star \in \underset{k \in \mathbb{R}^{L_{\min}}}{\text{argmax}}\ \text{Score}\left(k,\ \ldots,\ k,\ \mathbb{1}_p\right)\ \{L_{\min} = \min_{i \in \mathbb{R}^n} L_i\}$
Initial guess : $x^0 = \left(k^\star, \ldots, k^\star, \mathbb{1}_p\right)$
while $|\text{Score}(x^i) - \text{Score}(x^{i-1})| \geq \varepsilon_1$ and $i \leq \text{Iter max}_1$ **do**
  $x_1^i \in \underset{j \in \mathbb{R}^{L_1}}{\text{argmax}}\ \text{Score}\left(j,\ x_2^{i-1},\ x_3^{i-1},\ \ldots\ ,\ x_n^{i-1}, \mathbb{1}_p\right)$
  ...
  $x_n^i \in \underset{j \in \mathbb{R}^{L_n}}{\text{argmax}}\ \text{Score}\left(x_1^i,\ x_2^i,\ x_3^i,\ \ldots\ , j, \mathbb{1}_p\right)$
  i += 1
end while

Full coordinate ascent optimization
Use previous solutions: $X^* = (x_1^i, \ldots, x_n^i, \mathbb{1}_p)$ {i is the last index in previous while loop}
$Y^* = \text{Score}\,(X^*)$
while $|Y - Y^*| \geq \varepsilon_2$ and iteration $\leq \text{Iter max}_2$ **do**
  for i=1 . . . N **do**
    $X = X^*$
    $X_i = \text{not}\left(X_i^*\right)$ {not(0) = 1 and not(1) = 0}
    if $\text{Score}\,(X) \geq \text{Score}\,(X^*)$ **then**
      $X^* = X$
    end if
  end for
  $Y = \text{Score}\,(X^*)$
  iteration += 1
end while
Return X^*, Y^*

Assumption 1.5.1. We assume our objective function f is twice differentiable and strongly convex with respect to the Euclidean norm:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\sigma}{2} \|y - x\|_2^2 \tag{1.1}$$

for some $\sigma > 0$ and any $x, y \in \mathbb{R}^n$. We also assume that each gradient's coordinate is uniformly L_i Lipschitz, that is, there exists a constant L_i such that for any $x \in \mathbb{R}^n, t \in \mathbb{R}$

$$\left|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i\right| \leq L_i |t| \tag{1.2}$$

We denote by $L_{\max}$ the maximum of these Lipschitz coefficients :

$$L_{\max} = \underset{i=1...n}{max} L_i \tag{1.3}$$

We assume that the minimum of f denoted by $f^\star$ is attainable and that the left value of the epigraph with respect to our initial starting point x_0 is bounded, that is

$$\max_x \left\{ \left\| x - x^\star \right\| : f(x) \leq f(x_0) \right\} \leq R_0 \tag{1.4}$$

Remark 1.3. Strong convexity means that the function is between two parabolas. Condition 1.2 implies that the Gradient's growth is at most linear. Inequality 1.4 States that the function is increasing at infinity.

Proposition 1.4. *Under assumption 1.5.1, coordinate ascent optimization (cf. Algorithm 1) converges to the global minimum f^* at a linear rate proportional to $2nL_{\max}R_0^2$, that is*

$$\mathbb{E}[f(x_k)] - f^\star \leq \frac{2nL_{max}R_0^2}{k} \tag{1.5}$$

Proof. See Saltiel and Benhamou [2018] appendix A.1 first part of the proof. □

Proposition 1.5. *Under the same condition as proposition 1.4 and with $\sigma > 0$, we have another convergence rate that decreases exponentially fast as follows:*

$$\mathbb{E}[f(x_k)] - f^\star \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k (f(x_0) - f^\star) \tag{1.6}$$

Proof. See Saltiel and Benhamou [2018] appendix A.1 second part of the proof. □

In the case of a large σ, the second rate of convergence is much faster than the first one.
Our function to be maximized is obviously not convex. However, a linear rate in the convex case is rather a good performance for the ascent optimization method. Provided the method generalizes which is still under research, this convergence rate is a good hint of the efficiency of this method.

Table 1.1: Method Comparison: for each row, we provide in red the best(s) (hottest) method(s) and in blue the worst (coldest) method, while intermediate methods are in orange. We can notice that OCA achieves a higher score with the minimum feature sets. For the same feature set, RFE performs worse or equally, if we want the same performance for RFE, we need to have a larger feature set. BCA is the worst method both in terms of score and minimum feature set.

Method	OCA	RFE 24 features	BCA	RFE 28 features
% of features	16.6	16.6	27.08	19.4
Score (in %)	62.8	62.39	62.19	62.8

1.6 Experimental analysis

We present herein the result of the machine learning experiment with a real-life trading strategy. For full reproducibility, full data set and corresponding python code for this algorithm are available publicly on github with the limitation that sensitive data have been either anonymized or removed (like for instance the final pnl curve).

We first compare our method with two other state of the art methods: Recursive Feature Elimination (RFE) and Binary Coordinate Ascent (BCA) as presented in Zarshenas and Suzuki [2016].

Recursive Feature Elimination (RFE) (as presented in Mangal and Holm [2018]) first fits a model and removes features until a pre-determined number of features. Features are ranked through an external model that assigns weights to each features and RFE recursively eliminates features with the least weight at each iteration. One of the main limitation to RFE is that it requires the number of features to keep. This is hard to guess a priori and one may need to iterate much more than the desired number of feature to find an optimal feature set.

Binary Coordinate Ascent (BCA) is an iterative deterministic local optimization method to find Feature subset selection (FSS). The algorithm searches throughout the space of binary coded input variables by iteratively optimizing the objective function in each dimension at a time. Because there are no similarities used in the coordinate ascent optimization, it performs slowly compared to the OCA method.

On our test sets, we examine the accuracy score (the percentage of good classification). OCA method achieves the Pareto optimality as it reaches a score of 62.80 % with 16% of features used, to be compared to RFE that achieves 62.80 % with 19% of features used. BCA performs poorly with its highest score given by 62.19 % with 27% of features used. If we take in terms of efficiency criterium, the highest score with the least feature, OCA method is the most efficient among these three methods. In comparison, with the same number of features, namely 16%, RFE gets a score of 62.40 %. All these figures are summarized in the table 1.1.

It is illuminating to look at the histogram of gains and losses of our trades over our 10 years of history. Not surprisingly, we can observe two peaks corresponding to the profit target and stop loss level as shown in figure 1.2. This is quite obvious, but it is much better to use the pnl curve in the native currency of the underlying instrument than to look at the consolidated currency of our trading strategies to avoid foreign exchange noise as shown in figure 1.3.

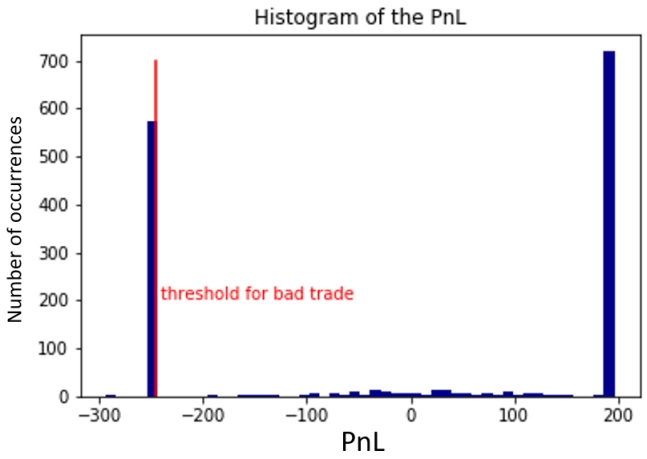


Figure 1.2: Histogram of the profit and loss (PnL) in Dollars amount (re-normalized for anonymity). We can observe two peaks corresponding to the profit target and stop loss levels. This is logical as the trading strategy examined here is a fixed profit target and stop loss strategy. As soon as a trade reaches these levels, the gain or loss is crystallized. If the market stays in the trading range and does not reach the level, we have a timeout in the strategy that cuts the strategy regardless of its pnl. These cases are rather rare and hence represent very small bars in the histogram.

1.7 Discussion

Compared to BCA our method reduces the number of iterations as it uses the fact that variables can be regrouped into categories or classes. Below is provided the number of iterations for OCA and BCA in figure 1.4. Our method requires only 350 iterations steps to converge as opposed to BCA which needs up to 700 iterations steps as it computes blindly variables ignoring similarities between the different variables.

Graphically, we can compute the best candidates for the four methods listed in table 1.1 in figure 1.5 and 1.6. We have taken the following color code. The hottest (or best performing) method is plotted in red, while the worst is in blue. Average performing methods are plotted in orange. In order to compare finely OCA and RFE, we have plotted in figure 1.6 the result of RFE for used features set percentage from 10 to 30 percent. We can notice that for the same feature set as OCA, RFE has a lower score, and equally that to get the same score as OCA, RFE needs a large features set.

We then look at the final goal which is to compare the trading strategy with and without machine learning. A standard way in machine learning is to split our data set between a randomized training and test set. We keep one third of our data for testing to spot any potential overfitting. If we use the standard and somehow naive way to take randomly one third of the data for our test set, we break the time dependency of our data. This has two consequences. We use in our training set some data that are after our test sets which is not realistic compared to real life. We also neglect any regime change in our data by mixing data that are not from the same period of time. However, we can do the test on this mainstream approach and compare the trading strategy with and without machine learning filtering. This is provided in figure 1.7. Since the blue curve that represents the combination of our algorithmic trading strategy and the oca method is above the orange one,

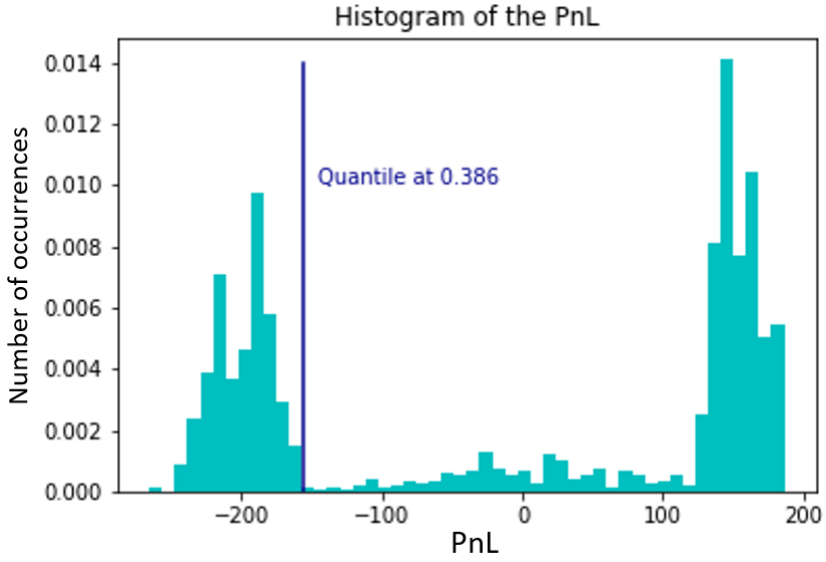


Figure 1.3: Same Histogram of the PnL but in Euros. Although it may seem very basic, it is important to use the native currency of the algorithmic trading strategy to avoid currency noise. Compared to figure 1.2, the only difference is to observe the profit and loss not in dollars but in euros as we consolidate all our trading strategies in euros. This is not a good practice as it introduces some additional noise in our labels as the Euro Dollar fx rate randomizes slightly the pnl outcome and hence some time-out exit may be confused with some bad exits.

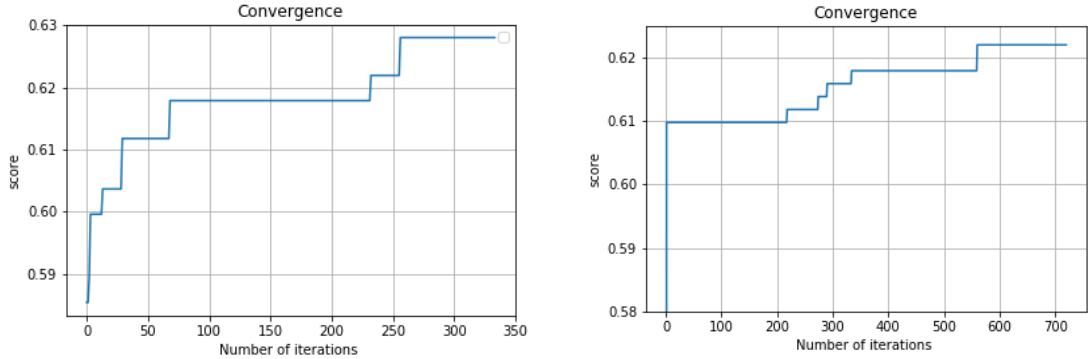


Figure 1.4: Iterations steps up to convergence for OCA and BCA. We represent the accuracy score in y-axis and the number of iterations of our algo in x-axis. OCA method is on the left while BCA is on the right. We see that OCA requires around 350 iteration steps to converge while BCA requires double around 700 iteration steps to converge

we experimentally validate that using machine learning enhances the overall profitability of our trading strategy by avoiding bad trades.

If instead, we split our set into two sets that are continuous in time, meaning we use as a training test the first two third of the data when there are sorted in time and as a test set the last third of

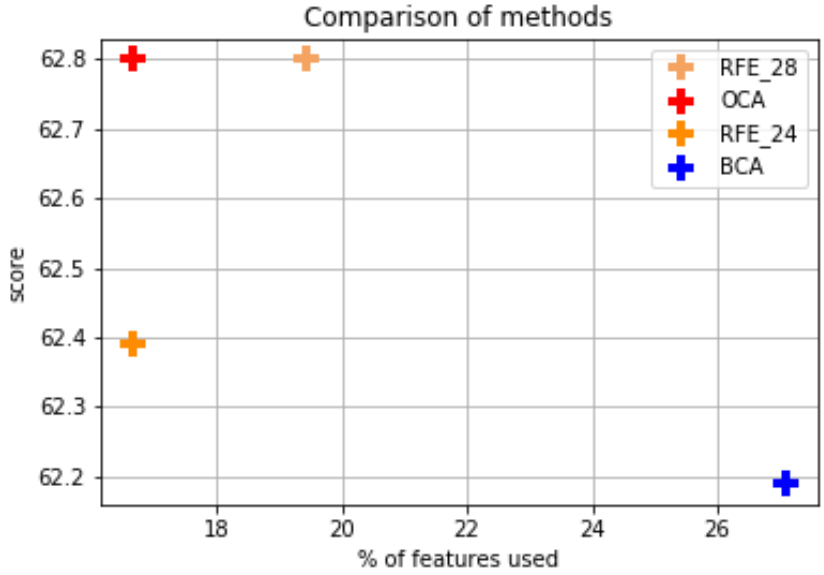


Figure 1.5: Comparison between the 4 methods. To qualify the best method, it should be in the upper left corner. The desirable feature is to have as few features as possible and the highest score. We can see that the red cross that represents OCA is the best. The color code has been designed to ease readability. Red is the best, orange is a slightly lower performance while blue is the worst.

the data, we get better result as the divergence between the blue and orange curve is larger. An explanation of this better efficiency may come from the fact that the non-randomization of the training set makes the learning for our model easier and leads to less overfitting overall. The method of splitting the two sets: training and test set into two sets rely on a temporal split, hence the title of our figure 1.8.

Last but not least, we can zoom the two curves when taking the test set with a temporal split. We clearly see that the method performs well to avoid selecting bad trades and hence the blue line decreases less than the orange one as shown in figure 1.9.

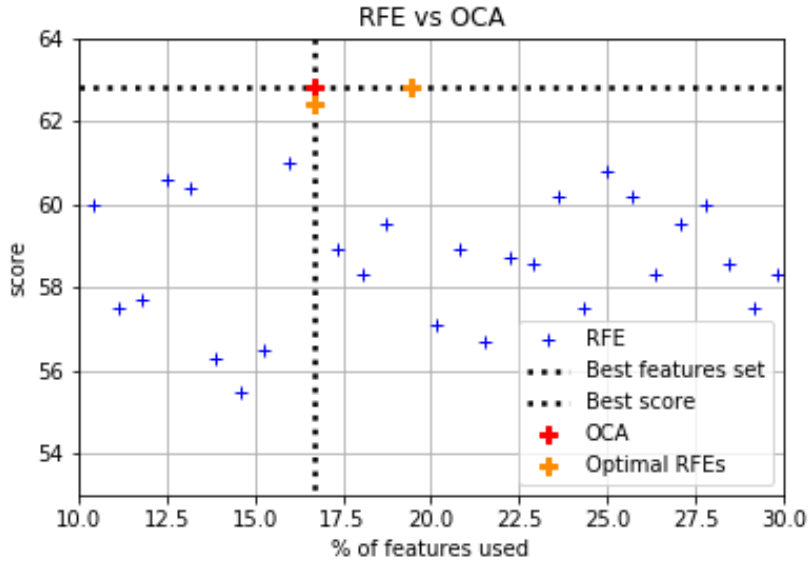


Figure 1.6: Comparison between OCA and RFE. Zoom on the methods. For RFE, we provide the score for various features set in blue. The two best RFE performers points are the orange cross marker points that are precisely the ones listed in table 1.1. The red cross marker point represents OCA. It achieves the best efficiency as it has the highest score and the smallest feature set for this score.

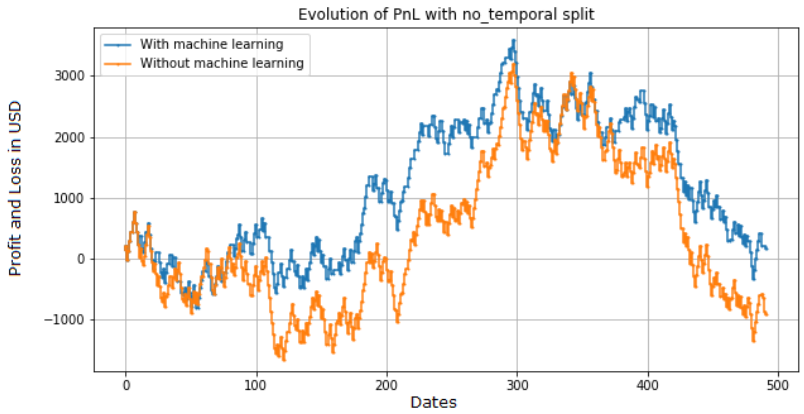


Figure 1.7: Evolution of the PnL with a randomized test set. The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the oca method to train our xgboost method

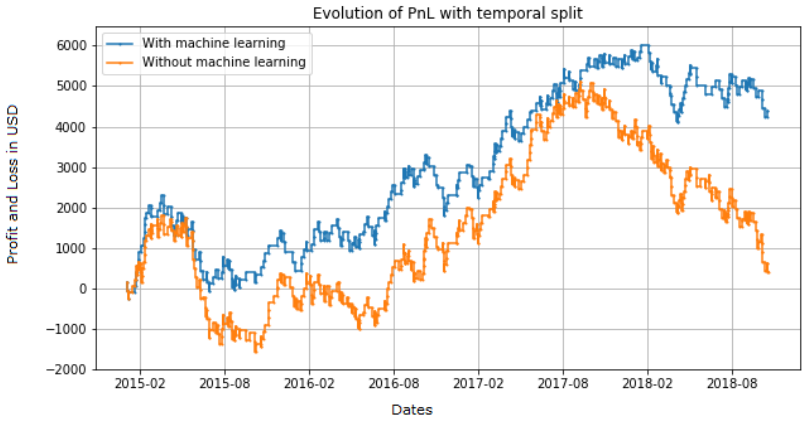


Figure 1.8: Evolution of the PnL with a test set given by the last third of the data to take into account temporality in our data set. The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the OCA method to train our xgboost method

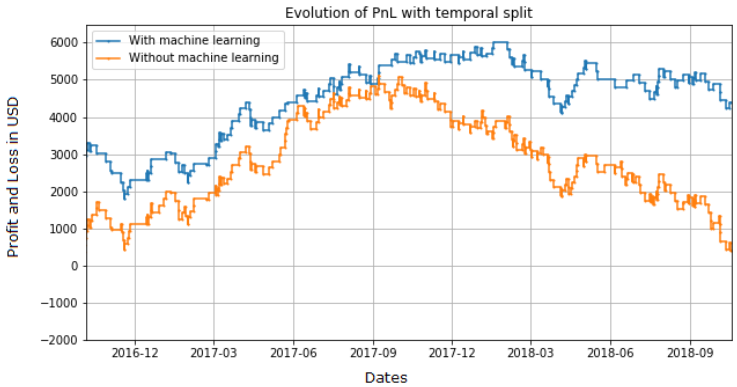


Figure 1.9: Zoom of the evolution of the PnL with a temporal split.The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the OCA method to train our xgboost method

1.8 Summary

In this chapter, we present an initial work of the thesis on selecting trades generated by an algorithmic trading method. We show that the algorithmic trading method can be enhanced with supervised learning methods. The challenge is to use measurements and information regrouped into features to detect before orders are electronically sent to the exchange highly probable non-successful trades. Because the logic of the algorithmic trading strategy may be challenging to understand, an agnostic supervised learning method can come to the rescue. However, choosing the best features in our initial features set is tricky as more data simultaneously provides additional information and noise at the same time. We present here OCA, a new feature selection method that leverages similarities between features. This method is not very demanding in terms of features knowledge and can efficiently select the best features without testing all possible features sets. This makes the complexity of the features selection problem polynomial. When implemented on real case strategies, we can empirically validate that the supervised learning method enhances overall trading profitability. As we ask the algorithm to detect in pre-trade operations highly unsuccessful candidates, the method is logically able to reduce overall draw-downs. The method developed herein is quite general and can be applied to any general supervised learning binary classification. In further work, we would like to explore reinforcement learning methods to adjust our method for capacity constraints as this is a limitation of the supervised learning approach. In the next chapter, we will extend the approach and use machine learning to select volatility targeting models.

The idea and concepts of this chapter have been presented in the conference Risk Forum 2019 as a full paper Saltiel and Benhamou [2019].

2 Adaptive learning for financial markets

2.1 Motivations

Model-Free Reinforcement Learning has achieved meaningful results in stable environments but, to this day, it remains problematic in regime-changing environments like financial markets. In contrast, model-based RL is able to capture some fundamental and dynamic concepts of the environment but suffers from cognitive bias. In this work, we propose to combine the best of the two techniques by selecting various model-based approaches thanks to Model-Free Deep Reinforcement Learning. Using not only past performance and volatility, we include additional contextual information such as macro and risk appetite signals to account for implicit regime changes. We also adapt traditional RL methods to real-life situations by considering only past data for the training sets. Hence, we cannot use future information in our training data set as implied by K-fold cross-validation. Building on traditional statistical methods, we use the traditional "walk-forward analysis", which is defined by successive training and testing based on expanding periods, to assert the robustness of the resulting agent.

Finally, we present the concept of statistical difference's significance based on a two-tailed T-test, to highlight the ways in which our models differ from more traditional ones. Our experimental results show that our approach outperforms traditional financial baseline portfolio models such as the Markowitz model in almost all evaluation metrics commonly used in financial mathematics, namely net performance, Sharpe and Sortino ratios, maximum drawdown, maximum drawdown over volatility.

2.2 Introduction

Reinforcement Learning (RL) aims at the automatic acquisition of skills or some other form of intelligence, to behave appropriately and wisely in comparable situations and potentially in situations that are slightly different from the ones seen in training. When it comes to real-world situations, there are two challenges: having a data-efficient learning method and being able to handle complex and unknown dynamical systems that can be difficult to model and are too far away from the systems observed during the training phase. Because the dynamic nature of the

environment may be challenging to learn, the first stream of RL methods has consisted in modeling the environment with a model. Hence it is called model-based RL. Model-based methods tend to excel in learning complex environments like financial markets. In mainstream agents literature, examples include robotics applications, where it is highly desirable to learn using the lowest possible number of real-world trials Kaelbling et al. [1996]. It is also used in finance where there are a lot of regime changes Freitas et al. [2009]. Niaki and Hoseinzade [2013] used artificial neural networks to forecast the level of the S&P 500 index. Heaton et al. [2017] used deep learning hierarchical models to solve problems in financial prediction and classification. Hence, the first generation of model-based RL, relying on Gaussian processes and time-varying linear dynamical systems, provides excellent performance in low-data regimes Deisenroth et al. [2011]. Deisenroth and Rasmussen [2011a] take a model-based and data-efficient approach to do policy search for robotics. Later, in a companion paper, Deisenroth et al. [2014] used Gaussian processes for data-efficient learning in robotics and control. Levine and Koltun [2013] shows that guided policy which is a type of model-based reinforcement learning works well while Kumar et al. [2016] experiments optimal control with learned local models for dexterous manipulation with success. A second generation, leveraging deep networks Gal et al. [2016] orDepeweg et al. [2016] has emerged and is based on the fact that neural networks offer high-capacity function approximators even in domains with high-dimensional observations Ebert et al. [2018], Kaiser et al. [2019] while retaining some sample efficiency of a model-based approach. Recently, it has been proposed to adapt model-based RL via meta policy optimization to achieve asymptotic performance of model-free models Clavera et al. [2018]. For a full survey of the model-based RL model, please refer to Moerland et al. [2020]. In finance, it is common to scale portfolio's allocations based on volatility and correlation as volatility is known to be a good proxy for the level of risk and correlation a standard measure of dependence. It is usually referred to as volatility targeting. It enables the portfolio under consideration to achieve close to constant volatility through various market dynamics or regimes by simply sizing the portfolio's constituents according to volatility and correlation forecasts.

In contrast, the model-free approach aims to learn the optimal actions blindly without a representation of the environment dynamics. Works like Mnih et al. [2015], Lillicrap et al. [2016], Haarnoja et al. [2018] have come with the promise that such models learn from raw inputs (and raw pixels) regardless of the game and provide some exciting capacities to handle new situations and environments, though at the cost of data efficiency as they require millions of training runs.

Hence, it is not surprising that the research community has focused on a new generation of models combining model-free and model-based RL approaches. A first idea has been to combine model-based and model-free updates for Trajectory-Centric RL. Chebotar et al. [2017]. Another idea has been to use temporal difference models to have a model-free deep RL approach for model-based control Pong et al. [2018]. van Hasselt et al. [2019] answers the question of when to use parametric models in reinforcement learning. Likewise, Janner et al. [2019] gives some hints when to trust model-based policy optimization versus model-free. Feinberg et al. [2018] shows how to use model-based value estimation for efficient model-free RL.

All these studies mostly applied to robotics and virtual environments, have not hitherto been widely used for financial time series. Our aim is to be able to distinguish various financial models that can be read or interpreted as model-based RL methods. These models aim at predicting volatility in financial markets in the context of portfolio allocation according to volatility target methods. These models are quite diverse and encompass statistical models based on historical data such as simple and naive moving average models, multivariate generalized auto-regressive conditional

heteroskedasticity (GARCH) models, high-frequency based volatility models (HEAVY) Noureldin and Shephard [2012] and forward-looking models such as implied volatility or PCA decomposition of implied volatility indices. To be able to decide on an allocation between these various models, we rely on deep model-free RL. However, using just the last data points does not work in our cases as the various volatility models have very similar behaviors. Following Benhamou et al. [2020a] and Benhamou et al. [2021e], we also add contextual information like macro signals and risk appetite indices to include additional information in our DRL agent hereby allowing us to choose the pre-trained models that are best suited for a given environment.

2.2.1 Related works

The literature on portfolio allocation in finance using either supervised or reinforcement learning has been attracting more attention recently. Initially, Heaton et al. [2017] use deep networks to forecast next period prices and to use this prediction to infer portfolio allocations. The challenge of this approach is the weakness of predictions: financial markets are well known to be non-stationary and to present regime changes (see Dias et al. [2015] or Benhamou [2018]).

More recently, Jiang and Liang [2016] Liang et al. [2018] Ye et al. [2020] or Benhamou et al. [2021a, 2020b, 2021e, 2020a] have started using deep reinforcement learning to do portfolio allocation. Transaction costs can be easily included in the rules. However, these studies rely on very distinct time series, which is a very different setup from our specific problem. They do not combine a model-based with a model-free approach. In addition, they do not investigate how to rank features, which is a great advantage of methods in ML like decision trees. Last but not least, they never test the statistical difference between the benchmark and the resulting model.

2.2.2 Contribution

Our contributions are precisely motivated by the shortcomings presented in the aforementioned remarks. They are four-fold:

- **The use of model-free RL to select various models that can be interpreted as model-based RL**. In a noisy and regime-changing environment like financial time series, the practitioners' approach is to use a model to represent the dynamics of financial markets. We use a model-free approach to learn from states to actions and hence distinguish between these initial models and choose which model-based RL to favor. In order to augment states, we use additional contextual information.
- **The walk-forward procedure.** Because of the non stationary nature of time-dependent data, and especially financial data, it is crucial to test DRL model stability. We present a traditional methodology in finance but never used to our knowledge in DRL model evaluation, referred to as walk-forward analysis that iteratively trains and tests models on extending data sets. This can be seen as the analogy of cross-validation for time series. This allows us to validate that the selected hyper-parameters work well over time and that the resulting models are stable over time.
- **Features sensitivity procedure.** Inspired by the concept of feature importance in gradient boosting methods, we have created a feature importance of our deep RL model based on its sensitivity to features inputs. This allows us to rank each feature at each date to provide some explanations why our DRL agent chooses a particular action.

- **A statistical approach to test model stability.** Most RL papers do not address the statistical difference between the obtained actions and predefined baselines or benchmarks. We introduce the concept of statistical difference as we want to validate that the resulting model is statistically different from the baseline results.

2.3 Problem formulation

Asset allocation is a major question for the asset management industry. It aims at finding the best investment strategy to balance risk versus reward by adjusting the percentage invested in each portfolio's asset according to risk tolerance, investment goals and horizons.

Among these strategies, volatility targeting is very common. Volatility targeting forecasts the amount to invest in various assets based on their level of risk to target a constant and specific level of volatility over time. Volatility acts as a proxy for risk. Volatility targeting relies on the empirical evidence that a constant level of volatility delivers some added value in terms of higher returns and lower risk materialized by higher Sharpe ratios and lower drawdowns, compared to a buy and hold strategy Perchet et al. [2016] or Dreyer and Hubrich [2017]. Indeed it can be shown that Sharpe ratio makes a lot of sense for managers to measure their performance. The distribution of Sharpe ratio can be computed explicitly Benhamou [2019]. Sharpe ratio is not an accident and is a good indicator of manager performance Benhamou et al. [2019b]. It can also be related to other performance measures like Omega ratio Benhamou et al. [2019a] and other performance ratios Benhamou and Guez [2018]. It also relies on the fact that past volatility largely predicts future near-term volatility, while past returns do not predict future returns. Hence, volatility is persistent, meaning that high and low volatility regimes tend to be followed by similar high and low volatility regimes. This evidence can be found not only in stocks, but also in bonds, commodities and currencies. Hence, a common model-based RL approach for solving the asset allocation question is to model the dynamics of the future volatility.

To articulate the problem, volatility is defined as the standard deviation of the returns of an asset. Predicting volatility can be done in multiple ways:

- Moving average: this model predicts volatility based on moving averages.
- Level shift: this model is based on a two-step approach that allows the creation of abrupt jumps, another stylized fact of volatility.
- GARCH: a generalized auto-regressive conditional heteroske-dasticity model assumes that the return r_t can be modeled by a time series $r_t = \mu + \epsilon_t$ where μ is the expected return and ϵ_t is a zero-mean white noise, and $\epsilon_t = \sigma_t z_t$, where $\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$. The parameters $(\mu, \omega, \alpha, \beta)$ are estimated simultaneously by maximizing the log-likelihood.
- GJR-GARCH: the Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) model is a variation of the GARCH model (see Glosten et al. [1993]) with the difference that σ_t, the variance of the white noise ϵ_t, is modeled as: $\sigma_t^2 = \omega + (\alpha + \gamma_{t-1})\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$ where $I_{t-1} = 1$ if $r_{t-1} < \mu$ and 0 otherwise. The parameters $(\mu, \omega, \alpha, \gamma, \beta)$ are estimated simultaneously by maximizing the log-likelihood.
- HEAVY: the HEAVY model utilizes high-frequency data for the objective of multi-step volatility forecasting Noureldin and Shephard [2012].

- HAR: this model is an heterogeneous auto-regressive (HAR) model that aims at replicating how information actually flows in financial markets from long-term to short-term investors.
- Adjusted TYVIX: this model uses the TYVIX index to forecast volatility in the bond future market,
- Adjusted Principal Component: this model uses Principal Component Analysis to decompose a set of implied volatility indices into its main eigenvectors and renormalizes the resulting volatility proxy to match a realized volatility metric.
- RM2006: RM2006 uses a volatility forecast derived from an exponentially weighted moving average (EWMA) metric.

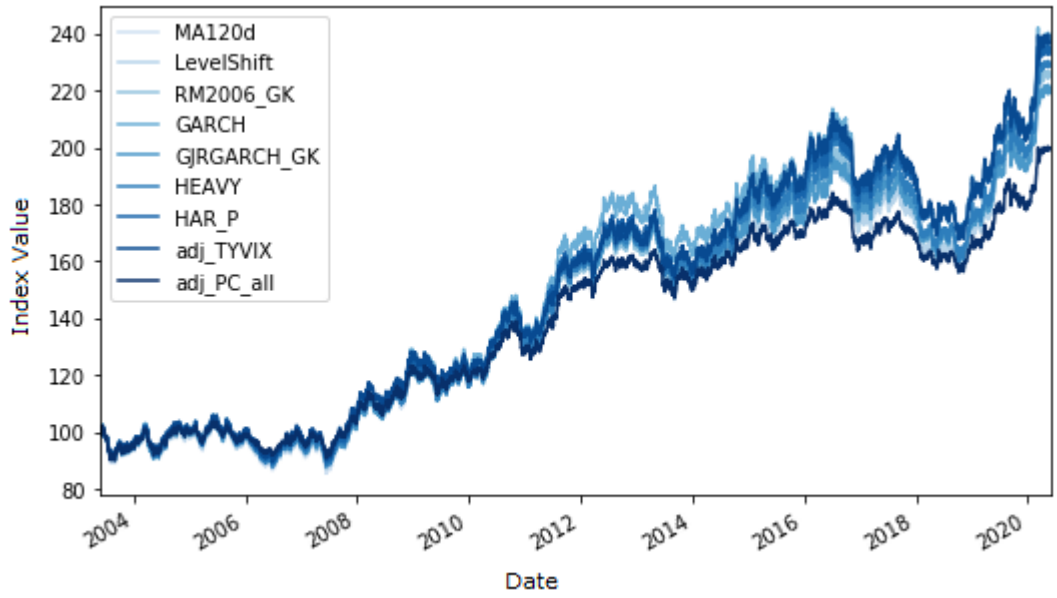


Figure 2.1: Volatility targeting model price evolution

2.3.1 Mathematical formulation

We have $n = 9$ models. Each model predicts a volatility for the rolled U.S. 10-year note future contract that we shall call "bond future" in the remainder of this chapter. The bond future's daily returns are denoted by r_t^{bond} and typically range from -2 to 2 percents with a daily average value of a few basis points and a daily standard deviation 10 to 50 times higher and ranging from 20 to 70 basis points. By these standards, the bond future's market is hard to predict and has a lot of noise making its forecast a difficult exercise. Hence, using some volatility forecast to scale position makes a lot of sense. These forecasts are then used to compute the allocation to the bond future's models. Mathematically, if the target volatility of the strategy is denoted by σ_{target} and if the model i predicts a bond future's volatility $\sigma_{t-1}^{i,pred}$, based on information up to $t-1$, the allocation in the bond future's model i at time t is given by the ratio between the target volatility and the predicted volatility: $k_{t-1}^{i} = \frac{\sigma_{target}}{\sigma_{t-1}^{i,pred}}$.

Hence, we can compute the daily amounts invested in each of the bond future volatility models and create a corresponding time series of returns $r_t^i = k_{t-1}^i \times r_t^{bond}$, consisting of investing in the bond future according to the allocation computed by the volatility targeting model i. This provides n time series of compounded returns whose values are given by $P_t^i = \prod_{u=t_1 \dots t} \left(1 + r_u^i\right)$. Our RL problem then boils down to selecting the optimal portfolio allocation (with respect to the cumulative reward) in each model-based RL strategies a_t^i such that the portfolio weights sum up to one and are non-negative $\sum_{i=1..n} a_t^i = 1$ and $a_t^i \geq 0$ for any $i = 1..n$. These allocations are precisely the continuous actions of the DRL model. This is not an easy problem as the different volatility forecasts are quite similar. Hence, the n time series of compounded returns look almost the same, making this RL problem non-trivial. Our aim is, in a sense, to distinguish between the indistinguishable strategies that are presented in figure 2.1. More precisely, figure 2.1 provides the evolution of the net value of an investment strategy that follows the different volatility targeting models.

Compared to standard portfolio allocation problems, these strategies' returns are highly correlated and similar as presented by the correlation matrix 2.2, with the lowest correlation of 97%. The correlation is computed as the Pearson correlation over the full data set from 2004 to 2020.

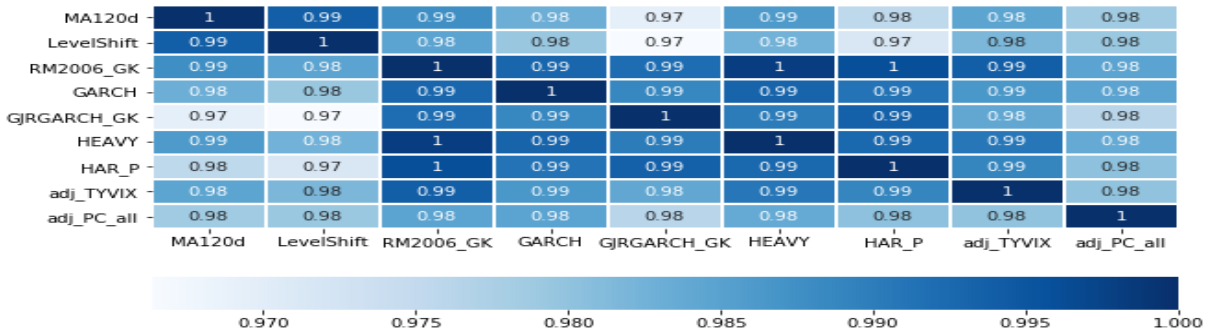


Figure 2.2: Correlation between the different volatility targeting models' returns

Following Sutton and Barto [2018], we formulate this RL problem as a Markov Decision Process (MDP) problem. We define our MDP with a 6-tuple $\mathcal{M} = (T, \gamma, \mathcal{S}, \mathcal{A}, P, r)$ where T is the (possibly infinite) decision horizon, $\gamma \in]0, 1]$ the discount factor, $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $p(s_{t+1}|s_t, a_t)$ the transition probability from the state s_t to s_{t+1} given that the agent has chosen the action a_t, and $r(s_t, a_t)$ the reward for a state s_t and an action a_t.

The agent's objective is to maximize its expected cumulative returns, given the start of the distribution. If we denote by π the policy mapping specifying the action to choose in a particular state, $\pi : \mathcal{S} \to \mathcal{A}$, the agent wants to maximize the expected cumulative returns. This is written as: $J^\pi = \mathbb{E}_{s_t \sim P, a_t \sim \pi} \left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \right]$.

MDP assumes that we know all the states of the environment and have all the information to make the optimal decision in every state.

From a practical standpoint, there are a few limitations to accommodate. First of all, the Markov property implies that knowing the current state is sufficient. Hence, we modify the RL setting by taking a pseudo-state formed with a set of past observations $(o_{t-n}, o_{t-n-1}, \dots, o_{t-1}, o_t)$. The

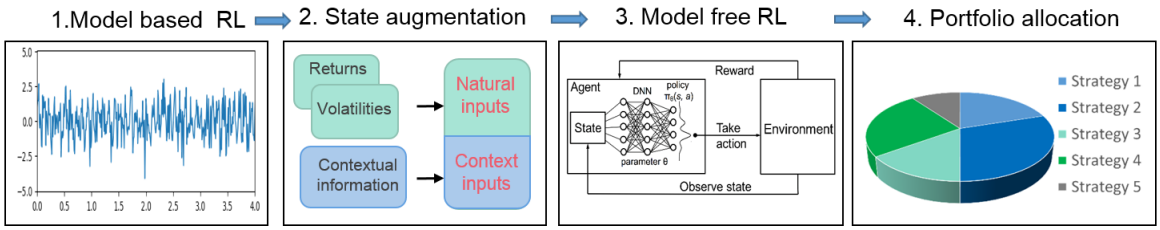


Figure 2.3: Overall architecture of our 4-steps model. We first have n models that represent the dynamics of the market volatility. We then add at step 2 the volatility and the contextual information to the states, thereby yielding augmented states. We then use at step 3 a model-free RL approach to find the portfolio allocation, at step 4, among the various volatility targeting models

trade-off is to take enough past observations to be close to a Markovian status without taking too many observations which would result in noisy states.

In our settings, the actions are continuous and consist in finding at time t the portfolio allocations a_t^i in each volatility targeting model. We denote by $a_t = \left(a_t^1, ..., a_t^n\right)^T$ the portfolio weights vector.

Likewise, we denote by $p_t = \left(p_t^1, ..., p_t^n\right)^T$ the closing price vector, and by $u_t = p_t \oslash p_{t-1} = \left(p_t^1/p_{t-1}^1, ..., p_t^n/p_{t-1}^n\right)^T$ the price relative difference vector, where $\oslash$ denotes the element-wise division,

and by $r_t = \left(p_t^1/p_{t-1}^1 - 1, ..., p_t^n/p_{t-1}^n - 1\right)^T$ the returns vector which is also the percentage change of each closing price $p_t^1, ..., p_t^n$. Due to price changes in the market, at the end of the same period, the weights evolve according to $w_{t-1} = (u_{t-1} \odot a_{t-1})/(u_{t-1}.a_{t-1})$ where $\odot$ is the element-wise multiplication, and . is the scalar product.

The goal of the agent at time t is hence to reallocate the portfolio vector from w^{t-1} to a_t by buying and selling the relevant assets, taking into account the transaction costs that are given by $\alpha|a_t - w_{t-1}|_1$ where α is the percentage cost for a transaction (which is quite low for future markets and given by 1 basis point) and $|.|_1$ is the L_1 norm operator. Hence at the end of time t, the agent receives a portfolio return given by $a_t.u_t - \alpha|a_t - w_{t-1}|_1$. The cumulative reward corresponds to the sum of the logarithmic returns of the portfolio strategy is given by $\mathbb{E}\left[\prod_{t=1}^T \log\left(a_t.u_t - \alpha|a_t - w_{t-1}|_1\right)\right]$, which is easier to process in a tensor flow graph as a log sum expression and is naturally given by $\mathbb{E}\left[\log\left(\sum_{t=1}^T a_t.u_t - \alpha|a_t - w_{t-1}|_1\right)\right]$.

Actions are modeled by a multi-input, multi-layer convolution network whose details are given by Figure 2.5. It has been shown that convolution networks are better for selecting features in portfolio allocation problem Benhamou et al. [2021b], Benhamou et al. [2021a] and Benhamou et al. [2020b]. The goal of the model-free RL method is to find the network parameters. This is done by an adversarial policy gradient method summarized by the algorithm 2 using traditional Adam optimization so that we have the benefit of adaptive gradient descent with root mean square propagation Kingma and Ba [2014] with a learning rate of 1% and 100,000 iteration steps with an early stop criterion if the cumulative reward does not improve after 15 full episodes. Because each episode is run on the same financial data, we use on purpose a vanilla policy gradient algorithm to

take advantage of the stability of the environment rather than use more advanced DRL agents like TRPO, DDPG or TD3 that would add on top of our model-free RL layer some extra complexity and noise.

Algorithm 2 Adversarial Policy Gradient

Input: initial policy parameters θ, empty replay buffer $\mathcal{D}$
repeat
  Reset replay buffer
  while not Terminal **do**
    Observe observation o and select action $a = \pi_\theta(o)$ with probability p and random action with probability $1 - p$,
    Execute a in the environment
    Observe next observation o', reward r, and done signal d to indicate whether o' is terminal
    Apply noise to next observation o'
    Store (o, a, o') in replay buffer $\mathcal{D}$
    if Terminal **then**
      for however many updates in $\mathcal{D}$ **do**
        Compute final reward R
      end for
      Update network parameter with Adam gradient ascent $\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}})$
    end if
  end while
until Convergence

2.3.2 Benchmarks

Markowitz

To benchmark our DRL approach, we need to compare it to traditional financial methods. Markowitz allocation as presented in Markowitz [1952a] is a widely-used benchmark in portfolio allocation as it is a straightforward and intuitive mix between performance and risk. In this approach, the risk is represented by the variance of the portfolio. Hence, the Markowitz portfolio minimizes variance for a given expected return, which is solved by standard quadratic programming optimization. If we denote by $\mu = (\mu_1, ..., \mu_n)^T$ the expected returns for our n model strategies and by Σ the covariance matrix of these strategies' returns, and by r_{min} the targeted minimum return, the Markowitz optimization problem reads

$$\begin{aligned} &\text{Minimize} \quad w^T \Sigma w \\ &\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1...l} w_i = 1, w \geq 0 \end{aligned}$$

Average

Another classical benchmark model for indistinguishable strategies is the arithmetic average of all the volatility targeting models. This seemingly naive benchmark is indeed performing quite well

as it mixes diversification and mean reversion effects.

Follow the winner

Another common strategy is to select the best performer of the past year and use it in the subsequent year. It replicates the standard investor's behavior that selects strategies that have performed well in the past.

2.3.3 Procedure and walk forward analysis

The whole procedure is summarized by Figure 2.3. We have n models that represent the dynamics of the market volatility. We then add the volatility and the contextual information to the states, thereby yielding augmented states. The latter procedure is presented as the second step of the process. We then use a model-free RL approach to find the portfolio allocation among the various volatility targeting models, corresponding to steps 3 and 4. To test the robustness of our resulting DRL model, we introduce a new methodology called walk-forward analysis.

Walk forward analysis

In machine learning, the standard approach is to do k-fold cross-validation. This approach breaks the chronology of data and potentially uses past data in the test set. Rather, we can take a sliding test set and take past data as training data. To ensure some stability, we favor adding incrementally new data in the training set, at each new step.

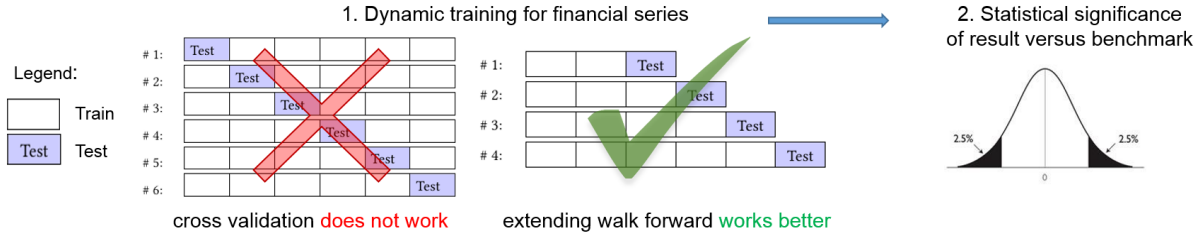


Figure 2.4: Overall training process: we use an "anchored walk forward" as we have anchored training data. In practice, and for our given data set, we train our models from date t to the end of t+1 and use a repetitive test period of one year from t+2 onward. Once a model has been selected, we also test its statistical significance, defined as the difference between the returns of two-time series.

This method is sometimes referred to as "anchored walk forward" as we have anchored training data. Finally, as the test set is always after the training set, the walk-forward analysis gives fewer steps compared with cross-validation. In practice, and for our given data set, we train our models from 2000 to the end of 2013 (giving us at least 14 years of data) and use a repetitive test period of one year from 2014 onward. Once a model has been selected, we also test its statistical significance, defined as the difference between the returns of two time series. We therefore do a T-test to validate how different these time series are. The whole process is summarized by Figure 2.4.

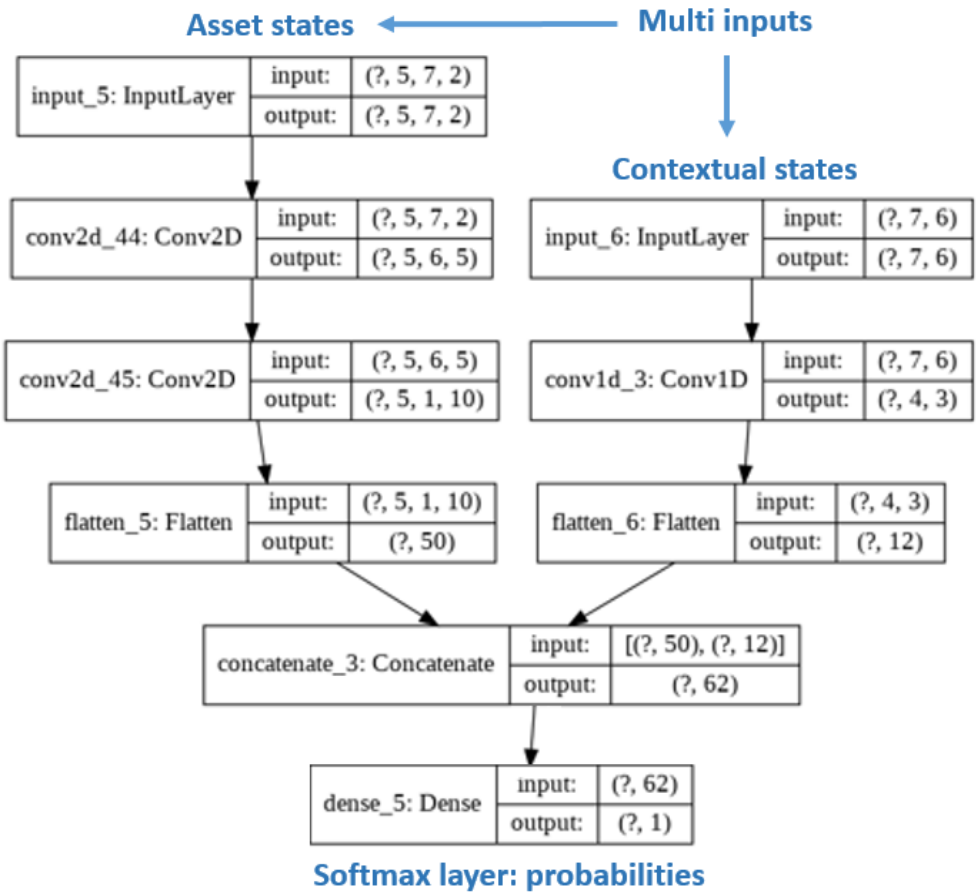


Figure 2.5: Multi-input DRL network. We use contextual inputs like short and long-term risk appetite indices and macro signals, portfolio strategy returns, and volatilities. We then have asset inputs such as the standard deviation of our assets to detect regime changes. We use a multi-input network with various convolutional layers and a final softmax layer to provide various allocations.

Model architecture

The states consist of two different types of data: the asset inputs and the contextual inputs.

Asset inputs are a truncated portion of the time series of financial returns of the volatility targeting models and the volatility of these returns computed over a period of 20 observations. So if we denote by r_t^i the returns of model i at time t, and by σ_t^i the standard deviation of returns over the last $d = 20$ periods, asset inputs are given by a 3-D tensor denoted by $A_t = [R_t, V_t]$, with

$$R_t = \begin{pmatrix} r_{t-n}^1 \ \ldots \ r_t^1 \\ \ldots \ \ldots \ \ldots \\ r_{t-n}^m \ \ldots \ r_t^m \end{pmatrix} \text{ and } V_t = \begin{pmatrix} \sigma_{t-n}^1 \ \ldots \ \sigma_t^1 \\ \ldots \ \ldots \ \ldots \\ \sigma_{t-n}^m \ \ldots \ \sigma_t^m \end{pmatrix}$$

This setting with two layers (past returns and past volatilities) is very different from the one presented in Jiang and Liang [2016], Zhengyao et al. [2017], Liang et al. [2018] that uses layers representing open, high, low, and close prices, which are not necessarily available for volatility target models. Adding volatility is crucial to detect regime change and is surprisingly absent from these works.

Contextual inputs are a truncated portion of the time series of additional data that represent contextual information. Contextual information enables our DRL agent to learn the context, and are, in our problem, short-term and long-term risk appetite indices and short-term and long-term macro signals. Additionally, we include the maximum and minimum portfolio strategies' return and the maximum portfolio strategies' volatility. Similarly to asset inputs, standard deviations is useful to detect regime changes. Contextual observations are stored in a 2D matrix denoted by C_t with stacked past p individual contextual observations. The contextual state reads

$$C_t = \begin{pmatrix} c_{t-n}^1 \ \ldots \ c_t^1 \\ \ldots \ \ldots \ \ldots \\ c_{t-n}^p \ \ldots \ c_t^p \end{pmatrix}.$$

The output of the network is a softmax layer that provides the various allocations. As the dimensions of the assets and the contextual inputs are different, the network is a multi-input network with various convolutional layers and a final softmax dense layer as represented in Figure 2.5.

Features sensitivity analysis

One of the challenges of neural networks relies on the difficulty to provide explainability about their behaviors. Inspired by computer vision, we present a methodology here that enables us to relate features to action. This concept is based on features sensitivity analysis. Simply speaking, our neural network is a multi-variate function. Its inputs include all our features, strategies, historical performances, standard deviations, contextual information, short-term and long-term macro signals, and risk appetite indices. We denote these inputs by X, which lives in $\mathbb{R}^k$ where k is the number of features. Its outputs are the action vector Y, which is an n-d array with elements between 0 and 1. This action vector lives in an image set denoted by $\mathcal{Y}$, which is a subset of $\mathbb{R}^n$. Hence, the neural network is a function $\Phi : \mathbb{R}^k \to \mathcal{Y}$ with $\Phi(X) = Y$. To project the various partial derivatives, we take the L1 norm (denoted by $|.|_1$) of the different partial derivatives as follows:

$|\frac{\partial \Phi(X)}{\partial X}|_1$. The choice of the L1 norm is arbitrary but is intuitively motivated by the fact that we want to scale the distance of the gradient linearly.

In order to measure the sensitivity of the outputs, simply speaking, we change the initial feature by its mean value over the last d periods. This is inspired by a "what if" analysis where we would like to measure the impact of changing the feature from its mean value to its current value. In computer vision, the practice is not to use the mean value but rather to switch off the pixel and set it to the black pixel. In our case, using a zero value would not be relevant as this would favor large features. We are really interested here in measuring the sensitivity of our actions when a feature deviates from its mean value.

The resulting value is computed numerically and provides us for each feature a feature importance. We rank these features importance and assign arbitrarily the value 100 to the largest and 0 to the lowest. This provides us with the following features importance plot given below 2.6. We can notice that the HAR returns and volatility are the most important features, followed by various returns and volatility for the TYVIX model. Although returns and volatility are dominating among the most important features, macro signals 0d observations come as the 12th most important feature over 70 features with a very high score of 84.2. The features sensitivity analysis confirms two things: i) it is useful to include volatility features as they are good predictors of regime changes, ii) contextual information plays a role as illustrated by the macro signal.

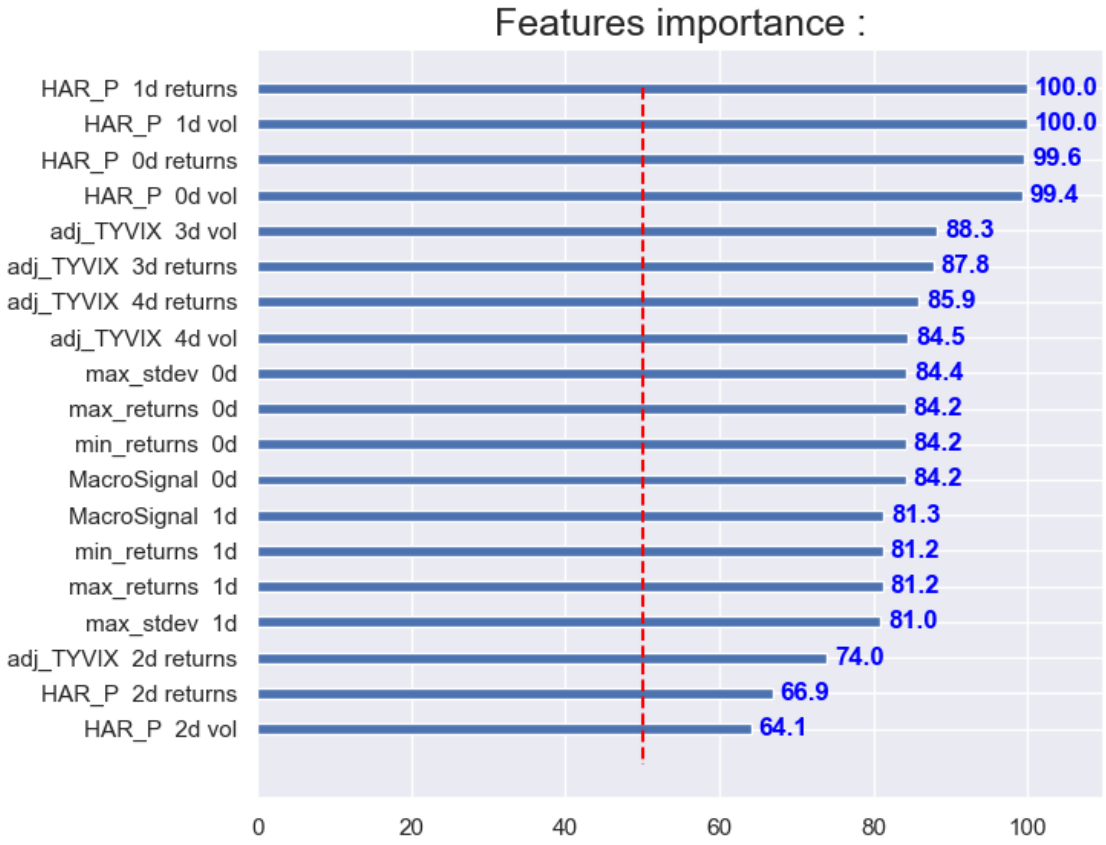


Figure 2.6: Features sensitivity analysis using features importance. We measure the impact changing the initial feature by its mean value over the last d periods

2.4 Out of sample results

In this section, we compare the various models: the deep RL model (DRL1) using states with contextual inputs and standard deviation, the deep RL model without contextual inputs and standard deviation (DRL2), the average strategy, the Markowitz portfolio and the "the winner" strategy. The results are the combination of the 7 distinct test periods: each year from 2014 to 2020. The resulting performance is plotted in Figure 2.7. We notice that the deepRL model with contextual information and the standard deviation is substantially higher than the other models in terms of performance as it ends at 157, whereas other models (the deepRL with no context, the average, the Markowitz and "the winner" model) end at 147.6, 147.8, 145.5, 143.4 respectively.

To make such a performance, the DRL model needs to frequently rebalance between the various models (Figure 2.8) with dominant allocations in GARCH and TYVIX models (Figure 2.9).

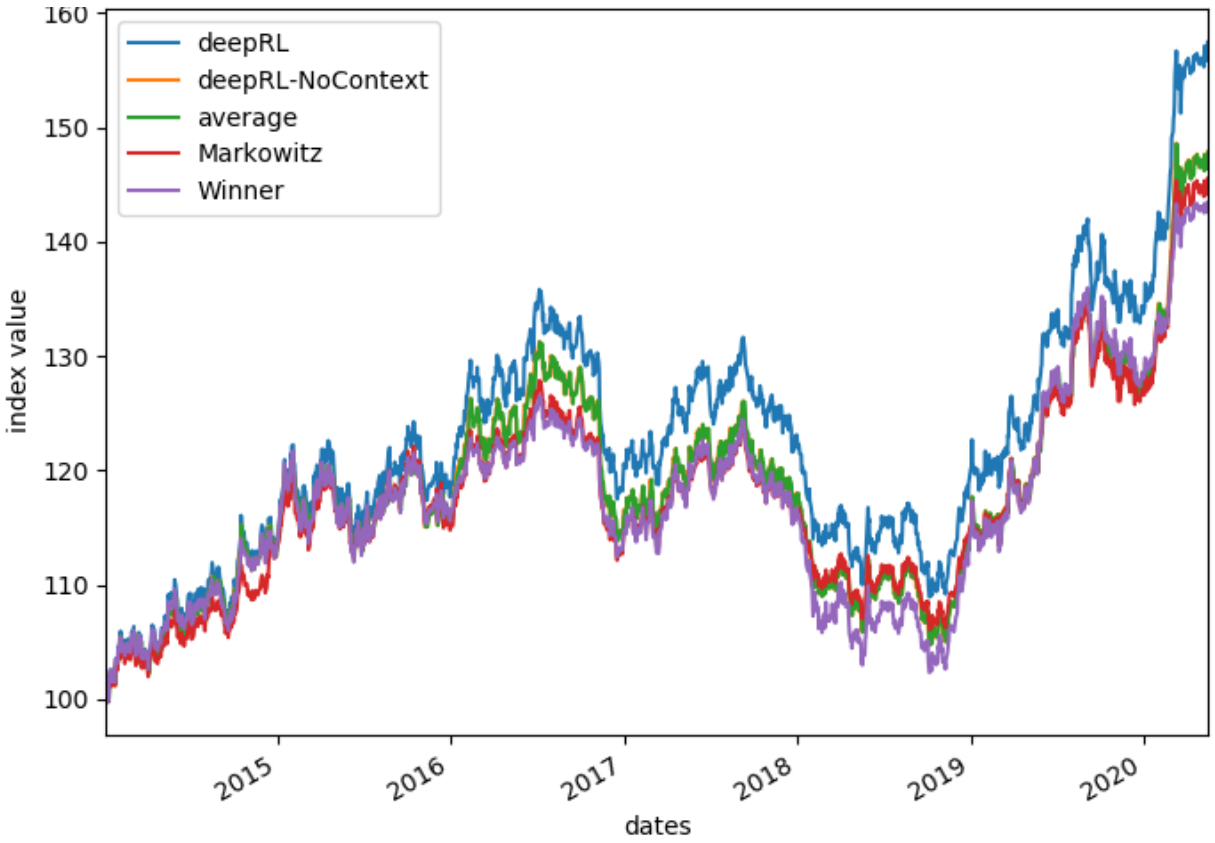


Figure 2.7: Model comparison

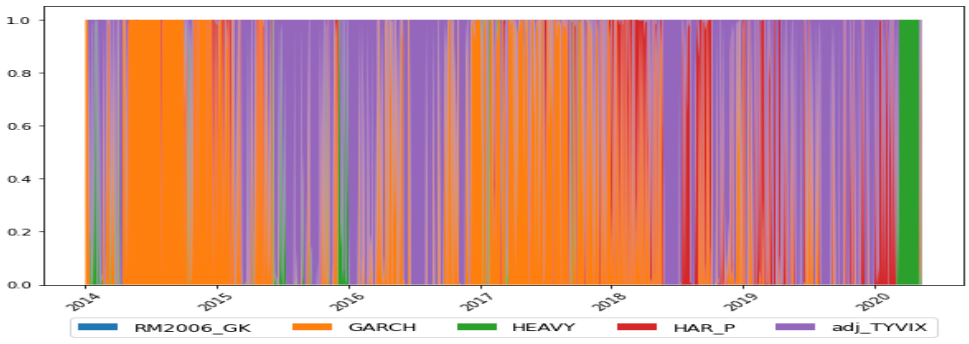


Figure 2.8: DRL portfolio allocation. We represent our DRL model allocation over time. The model is allocated on the best-predicted model based on its future returns. We notice the model doesn't reallocate often as it predicts one model to be the best for a long period of time such as GARCH model (orange one)

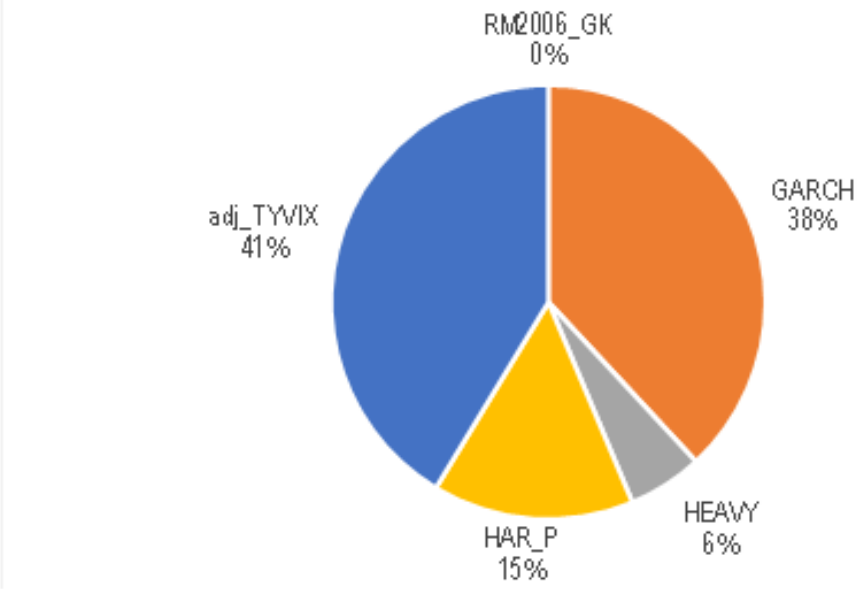


Figure 2.9: Average model allocation

2.4.1 Experimental analysis

Risk metrics

We provide various statistics in Table 2.1 for different time horizons: 1, 3 and 5 years. For each horizon, we put the best model, according to the column's criterion, in bold. The Sharpe and Sortino ratios are computed on daily returns. Maximum drawdown (written as mdd in the table), which is the maximum observed loss from a peak to a trough for a portfolio, is also computed on daily returns. DRL1 is the DRL model with standard deviations and contextual information, while DRL2 is a model with no contextual information and no standard deviation. Overall, DLR1, the DRL model with contextual information and standard deviation, performs better for 1, 3 and 5 years except for three-year maximum drawdown. Globally, it provides a 1% increase in annual net return for a 5-year horizon. It also increases the Sharpe ratio by 0.1 and is able to reduce most of the maximum drawdowns except for the 3-year period. Markowitz portfolio selection and "the winner" strategy, which are both traditional financial methods heavily used by practitioners, do not work that well compared with a naive arithmetic average and furthermore when compared to the DRL model with context and standard deviation inputs. A potential explanation may come from the fact that these volatility targeting strategies are very similar making the diversification effects non-effective.

Statistical significance

Following our methodology described in section 2.4, once we have computed the results for the various walk-forward test periods, we do a T-statistic test to validate the significance of the result. Given two models, we test the null hypothesis that the difference of the returns running average (computed as $(\sum_{u=0}^{t} r_u/t)$ for various times t) between the two models is equal to 0. We provide the T-statistic and, in parenthesis, the p-value. We take a p-value threshold of 5%, and put the

Table 2.1: Models comparison over 1, 3, 5 years

	return	sharpe	sortino	mdd	mdd/vol
			1 Year		
DRL1	**22.659**	**2.169**	**2.419**	**- 6.416**	**- 0.614**
DRL2	20.712	2.014	2.167	- 6.584	- 0.640
Average	20.639	2.012	2.166	- 6.560	- 0.639
Markowitz	19.370	1.941	2.077	- 6.819	- 0.683
Winner	17.838	1.910	2.062	- 6.334	- 0.678
			3 Years		
DRL1	**8.056**	**0.835**	**0.899**	- 17.247	**- 1.787**
DRL2	7.308	0.783	0.834	**- 16.912**	- 1.812
Average	7.667	0.822	0.876	- 16.882	- 1.810
Markowitz	7.228	0.828	0.891	- 16.961	- 1.869
Winner	6.776	0.712	0.754	- 17.770	- 1.867
			5 Years		
DRL1	**6.302**	**0.651**	**0.684**	**- 19.794**	**- 2.044**
DRL2	5.220	0.565	0.584	- 20.211	- 2.187
Average	5.339	0.579	0.599	- 20.168	- 2.187
Markowitz	4.947	0.569	0.587	- 19.837	- 2.074
Winner	4.633	0.508	0.526	- 19.818	- 2.095

cases where we can reject the null hypothesis in bold in table 2.2. Hence, we conclude that the DRL model with context (DRL1) model is statistically different from all other models. These results on the running average are quite intuitive as we are able to distinguish the DRL1 model curve from all other curves in Figure 2.7. Interestingly, we can see that the DRL model without context (DRL2) is not statically different from a pure averaging of the average model that consists in averaging allocation computed by model-based RL approaches.

Table 2.2: T-statistics and P-values (in parenthesis) for running average returns difference

Avg Return	DRL2	Average	Markowitz	Winner
DRL1	**72.1 (0%)**	**14 (0%)**	**44.1 (0%)**	**79.8 (0%)**
DRL2		1.2 (22.3%)	**24.6 (0%)**	**10 (0%)**
Average			**7.6(0%)**	0.9 (38.7%)
Markowitz				**-13.1 (0%)**

Financial interpretation

It is interesting to understand how the DRL model achieves such a performance as it provides an amazing additional 1% annual return over 5 years, and an increase in Sharpe ratio of 0.10. This is done simply by selecting the right strategies at the right time. This helps us to confirm that adaptive learning thanks to the model-free RL is somehow able to pick up regime changes. We notice that the DRL model selects the GARCH model quite often and, more recently, the HAR and HEAVY model (Figure 2.8). When targeting a given volatility level, capital weights are inversely proportional to the volatility estimates. Hence, lower volatility estimates give higher weights and in a bullish market give higher returns. Conversely, higher volatility estimates drive capital weights lower and have better performance in a bearish market. The allocation of these models evolves quite a bit as shown by Figure 2.10, which plots the rank of the first 5 models.

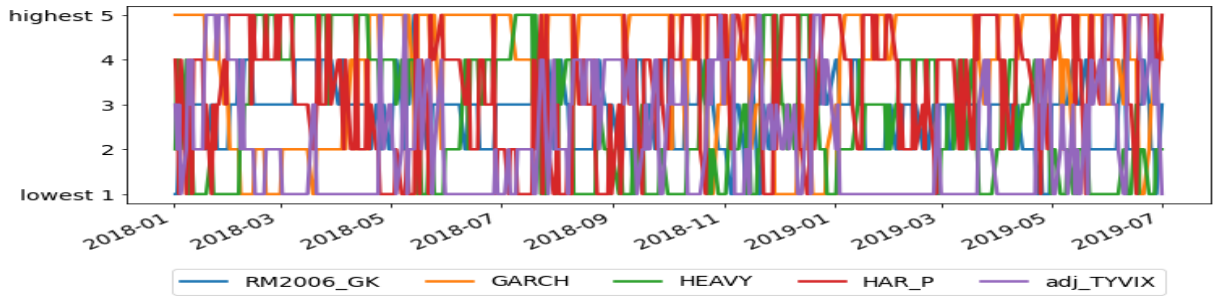


Figure 2.10: Volatility estimates rank. At each date, we compute the predicted rank of a model. If the rank is 5, it means that among the 5 potential models (namely RM2006_GK, GARCH, Heavy, HAR_P and Adj_TYVIX), the DRL model predicts that this model has the highest rank and is predicted to perform the best. Likewise, 1 is the lowest predicted rank for a model.

We can therefore test if the DRL model has a tendency to select volatility-targeting models that favor lower volatility estimates. If we plot the occurrence of rank by the dominant model for the

DRL model, we observe that the DRL model selects the lowest volatility estimate model quite often (38.2% of the time) but also tends to select the highest volatility models giving a U shape to the occurrence of rank as shown in figure 2.11. This U shape confirms two things: i) the model tends to select either the lowest or highest volatility estimates models, which are known to perform best in bullish markets or bearish markets (however, it does not select these models blindly as it is able to time when to select the lowest or highest volatility estimates); ii) the DRL model is able to reduce maximum drawdowns while increasing net annual returns as seen in Table 2.1. This capacity to simultaneously increase net annual returns and decrease maximum drawdowns indicate a capacity to detect regime changes. Indeed, a random guess would only increase the leverage when selecting the lowest volatility estimates, thus resulting in higher maximum drawdowns.

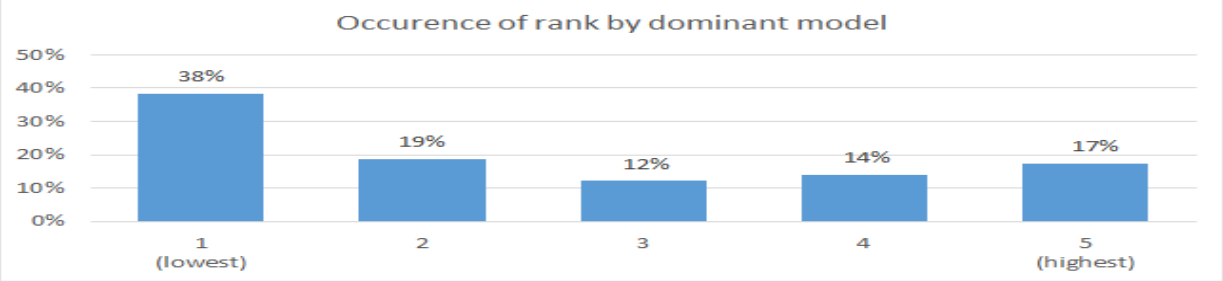


Figure 2.11: Occurrence of rank for the DRL model

2.4.2 Benefits of DRL

The advantages of context-based DRL are numerous: (i) by design, DRL directly maps market conditions to actions and can thus adapt to regime changes, (ii) DRL can incorporate additional data and be a multi-input method, as opposed to more traditional optimization methods.

2.4.3 Future work

As nice as this may look, there is room for improvement as more contextual data and architectural networks choices could be tested as well as other DRL agents like DDPG, TRPO or TD3. It is also worth mentioning that the analysis has been conducted on a single financial instrument and a relatively short out-of-sample period. Expanding this analysis further in the past would cover more various regimes (recessions, inflationary, growth, etc.) and potentially improve the statistical relevance of this study at the cost of losing relevance for more recent data. Another lead consists of applying the same methodology to a much wider ensemble of securities and identify specific statistical features based on distinct geographic and asset sectors.

2.5 Summary

In this chapter, we extend the work of the first chapter by proposing to create an adaptive learning method that combines model-based and model-free RL approaches to address volatility regime changes in financial markets. The model-based approach enables to efficiently capture the volatility dynamics while the model-free RL approach to time when to switch from one to another model. This combination enables us to have an adaptive agent that switches between different dynamics. We strengthen the model-free RL step with additional inputs like volatility and macro and risk appetite signals that act as contextual information. The ability of this method to reduce risk and profitability is verified when compared to the various financial benchmarks. The use of successive training and testing sets enables us to stress test the robustness of the resulting agent. Features sensitivity analysis confirms the importance of volatility and contextual variables and explains in part the DRL agent's better performance. Last but not least, statistical tests validate that results are statistically significant from a pure averaging method of all model-based RL allocations. This chapter concludes the first part on the model-based approach of machine learning for finance. The next part will describe what a model-free approach is and how to use it in finance.

This chapter has been the subject of two papers Benhamou et al. [2021d] and Benhamou et al. [2021c]. The second won the best paper award at the conference MIDAS 2021.

PART II

Model-free approaches

Overview

In contrast to part I, we will rely purely on Machine Learning to construct a dynamic portfolio without any initial quantitative model. This approach called model-free learning is formulated as a Reinforcement learning problem where the objective is to learn the mapping function between the initial financial states and the portfolio weights such that these weights maximize a reward function like the Sharpe ratio of the portfolio or its net performance over the training period.

The interest in relying on Reinforcement Learning (RL) to generate dynamic portfolios is at least twofold:

- As presented in chapter 3, we can adapt portfolio construction to different regimes and in particular to crisis environments like during the Covid period. While Reinforcement learning is obviously more sophisticated than Supervised Learning, the promise of RL through an intelligent agent that uses trial and error and improves its ability to achieve an objective based on rewards is to gain robustness and adapt to changing regimes.
- As presented in chapter 4, RL can generalize traditional Portfolio methods and reformulate them as a more general optimization problem where the portfolio method is indeed a mapping function between some financial variables called states in RL (the first two moments of the portfolio assets for instance in Markowitz) and the portfolio weights. This mapping function is the one that maximizes some reward functions (the Sharpe ratio). RL shows that portfolio methods that are convex optimization can be extended to more advanced and complex optimization that are in general nonconvex.

Furthermore, in this part, we show that Reinforcement Learning and Supervised Learning which are traditionally opposed are not that different and show that, in particular, the gradient policy method can be cast as a supervised learning problem where the true label is replaced with discounted rewards, emphasizing the tight connection between the two approaches. This is the subject of chapter 5.

3 Detecting Crisis with deep reinforcement learning

3.1 Motivations

Deep reinforcement learning (DRL) has reached superhuman levels in complex tasks like game solving (Go Silver et al. [2017], StarCraft II Vinyals et al. [2019]), and autonomous driving Wang et al. [2018]. However, it remains an open question whether DRL can reach a human level in applications to financial problems and in particular in detecting pattern crisis and consequently dis-investing. In this chapter, we present an innovative DRL framework consisting of two sub-networks fed respectively with portfolio strategies past performances, and standard deviations as well as additional contextual features. The second sub network plays an important role as it captures dependencies with common financial indicators features like risk aversion, economic surprise index, and correlations between assets that allow taking into account context based information. We compare different network architectures either using layers of convolutions to reduce network's complexity or LSTM block to capture time dependency and whether previous allocations are important in the modeling. We also use adversarial training to make the final model more robust. Results on the test set show this approach substantially over-performs traditional portfolio optimization methods like Markovitz and is able to detect and anticipate crisis like the current Covid one.

3.2 Introduction

Being able to adapt portfolio allocation to crisis environment like the current Covid crisis is a major concern for the financial industry. Indeed, the current Covid crisis took the industry by surprise twice. First, when stock markets plunged at an unprecedented speed in March 2020 with the SP 500 falling by 13 %, asset managers were slow to react and to cut risk exposure. And secondly, when stock markets bounced back up at an equally rapid pace, with a rise of 13 % for the SP 500 in May 2020, asset managers were again overhauled. In contrast, the previous 2008 crisis was very slow both in terms of its falls and recovery. Hence, adapting portfolio allocation to crisis environment is a very important matter and has attracted growing attention from the financial scientific community.

The standard approach for portfolio allocation, which serves as a baseline for our research, relies on determining portfolio weights according to a risk return criterion. The so-called Markowitz/ portfolio Markowitz [1952b] finds the optimal allocation by determining the portfolio with minimum variance given a target return or equivalently the portfolio with maximum return given a targeted level of variance (the dual optimization). However, this approach suffers from a major flaw because of unreliable risk estimations of the individual portfolio strategy excess returns and covariances. This leads not only to unstable allocations, but also to slow reactions to changing environments Black and Litterman [1992]. If we want to find a more dynamic allocation method, deep reinforcement learning is an appealing method. It reformulates the portfolio optimization problem as a continuous control program with delayed rewards. Rules are simple. Each trading day, the dynamic virtual asset manager agent has the right to modify the portfolio allocation. When it modifies the portfolio weights, it incurred transaction costs. The agent can only allocate between 0 and 100 % for all the portfolio assets. It can not short any asset, hence weights are always positive and never above 100 %. It can neither borrow to fund leverage positions, hence the sum of all allocations is strictly equal to 100 %. To make decisions, the dynamic agent has access not only to past performances but also to some financial contextual information that helps it make an informed decision. The agent receives in terms of feedback a financial reward that orientates its decisions. Compared to traditional financial methods, this approach has the major advantage to adapt to changing market conditions and being somehow more model free than traditional financial methods as we connect portfolio allocations directly to financial data and not to specific risk factors that may factor in some cognitive bias. This stream of research is also highly motivated by the recent major progress of deep reinforcement learning methods that have reached super human levels in complex tasks like game solving (historically Atari games Mnih et al. [2013], Go Silver et al. [2017], StarCraft II Vinyals et al. [2019]), and autonomous driving Wang et al. [2018]. Nonetheless, it still remains an open question whether DRL can reach a human level in applications to financial problems and in particular in detecting pattern crisis and consequently dis-investing.

3.2.1 Related Work

Initially, many of the machine-learning and in particular deep network applications to financial markets tried to predict price movements or trends Freitas et al. [2009], Niaki and Hoseinzade [2013], Heaton et al. [2017]. The logic was to take historical prices of assets as inputs and use deep neural networks to predict asset prices for the next period. Armed with the forecast, a trading agent can act and decide the best allocation. The problem to solve is a standard supervised learning task and more precisely a regression problem. It is straightforward to implement. Yet, the efficiency of the method relies heavily on the accuracy of the prediction, which makes the method quite fragile and questionable as future market prices are well known to be difficult to predict. Furthermore, this approach tends to reduce substantially portfolio diversification and can not cope easily with transaction costs. In contrast, DRL can easily tackle these issues as it does not aim at predicting prices but rather at finding the optimal action or for our matter the optimal allocation.

The idea of applying DRL to portfolio allocation has recently taken off in the machine learning community with some recent works on crypto currencies Jiang and Liang [2016], Zhengyao et al. [2017], Liang et al. [2018], Yu et al. [2019] and Wang and Zhou [2019]. Other works have focused on more common financial assets Benhamou et al. [2020c,a,b]. Compared to traditional approaches on financial time series, that aim at taking decision based on forecasting estimates,

Zhengyao et al. [2017] and Liang et al. [2018] showed that deep reinforcement learning with Convolutional Neural Network (CNN) architecture tends to perform better for crypto currencies and Chinese stock markets than deep learning architecture that relies on time series forecast like LSTM. However, when there is a very rapid crisis, like what happened during the Covid crisis, using just past performances may lead the DRL agent to react too slowly. To make an analogy, it is as if the agent was self-driving on the highway, and very brutally, an obstacle arises. Using past performances only is like looking in the mirrors behind to infer what will happen next. Adding a context is like lifting up our eyes and looking further forward. Context based reinforcement learning has recently emerged as a strong tool to increase reinforcement learning dynamic agent performance Gupta et al. [2018], Lee et al. [2020]. More specifically, context based reinforcement learning (RL) with high capacity function approximators, such as deep neural networks (DNNs), has in the last two years attracted growing attention and been the subject of many publications in notorious machine learning conferences as it solved efficiently a variety of sequential decision-making problems, including board games (e.g., Go and Chess Schrittwieser et al. [2019], video games like Atari games Kaiser et al. [2019]), and complex robotic control tasks Zhang et al. [2018], Nagabandi et al. [2018], Hafner et al. [2019]. Theoretically, it has also been advocated that the usage of a context enables achieving superior data-efficiency to model-free RL methods in general Deisenroth and Rasmussen [2011b], Levine and Abbeel [2014].

So in this work, we extend previous works of DRL by precisely using a context based approach. This is done by integrating common financial states in our deep network, having at least two sub networks and potentially three if we also incorporate in states the previous allocations. Experiments show that this approach is able to pick the best portfolio allocation out of sample using financial features used by asset managers: risk aversion index, correlation between equities and bonds, Citi economic surprise index and to accommodate for crisis by reducing risk exposure. We provide performances out of sample and test various configurations to emphasize that using CNN works much better than more predictive architecture like LSTM confirming previous works.

3.2.2 Contributions

Our contributions are twofold:

- First we explain why a context based deep reinforcement learning approach is closer to human thinking and leads to better results, with a novel deep network architecture consisting of two sub networks: one network (network 1) that takes as inputs past performances (and standard deviation) of the portfolio strategies and another one (network 2) that takes as inputs financial contextual information related to the performances of the portfolio strategies that are thought to have some predictive power regarding portfolio strategies future performances.

- Second, we summarize lots of empirical findings. The reward function is critical. Sharpe ratio reward leads to different results compared to a straight final net performance reward function. CNN performs better than LSTM and captures implicit features. Using adversarial training by adding noise to the data improves the model. Last but not least, dependency on previous allocations does not improve the model.

3.3 Mathematical formulation

As summarized by figure 3.1, an asset manager robot has several strategies that it wants to allocate optimally, with a performance objective on the overall portfolio. Not only does it have access to historical daily performance (the middle rectangle in figure 3.1) but it can also leverage additional information (the rectangle on the left in figure 3.1) that provides some contextual information about market conditions. These are other price data points but also unstructured data like some macro economic data. To gauge the performance of its decision, it has an objective that can be either the net performance of the portfolio or some risk return criterion (the third rectangle of figure 3.1 on the right)

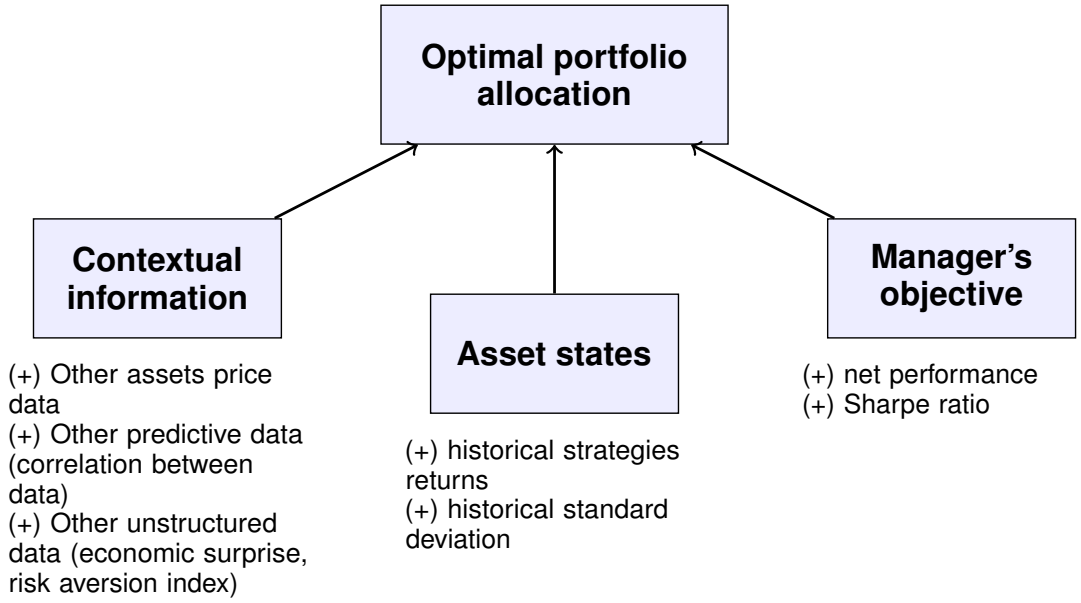


Figure 3.1: Portfolio allocation problem

The question of asset allocation can be reformulated as a standard reinforcement learning problem, thanks to Markov Decision Process (MDP). The learning agent interacts with an environment E to decide rational or optimal actions and receives in return some rewards. These rewards are not necessarily only positive and are given only at the end of the financial episode. These rewards act as feedback for finding the best action. Using the established formalism of Markov decision process, we assume that there exists a discrete time stochastic control process represented by a 4-tuple defined by $(\mathcal{S}, \mathcal{A}, P_a, R_a)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $P_a(s, s') = \mathbb{P}(s_{t+1} = s' \mid s_t = s, a_t = a)$ the transition probability that action a in state s at time t will lead to state s' at the next period $t + 1$ and finally, $R_a(s, a)$ the immediate reward received after state s and action a.

The requirement of a Markovian state that guarantees that there exists a solution (hence satisfying the Bellman optimality principle Sutton and Barto [2018]) is a strong assumption that is hard to verify in practice. It is somehow levied in practice by stacking enough observations to enforce that the Markov property is satisfied. Hence, it is useful, following Mnih et al. [2016] or Jaderberg et al. [2016], to introduce the concept of observations and pile them to coin states. In this setting,

the agent perceives at time t an observation o_t along with a reward r_t.

In our setting, time is divided into trading periods of equal length τ. In the rest of the chapter, τ represents one trading day but the setting can be applied to shorter time periods, like 30 minutes, to deal with intra-day trading decisions. At the beginning of each trading period, a trading robot decides to potentially reallocate the funds among m assets. The trading robot has access to an environment that provides at each time t, a state s_t that is composed of past observations that are rich enough to assume Markovianity. Intuitively, it is important for the agent to observe not only the last returns but also some previous returns (like the returns over 2, 3, and 4 business days, but also a week and potentially a month) to make a decision. Mathematically, we denote by δ_1 the lag operator applied to each observation. To make this concrete the lag operator δ_1 's outputs are the last portfolio strategy returns at time t but also at time $t-1,\ t-2,\ t-3$ and so on. There is here some trade-off. We obviously need enough observations to mimic a Markovian setting to ensure the problem is well posed. But we also need to reduce observations to avoid facing the curse of dimension. We will discuss this point in our experience, but practically, we take returns at time $t-60$ representing returns 3 months ago, $t-20$ one month ago[1], $t-4,\ t-3,\ t-2,\ t-1$ and t, the latter four providing returns over the last trading week. By abuse of language, we can represent the lag δ_1 operator by a vector of lagging periods $\delta_1 = [0, 1, 2, 3, 4, 20, 60]$ (as there is a one to one mapping between the operator and the lagging periods) and retrieve the corresponding returns for asset i as follows: $\left[r_t^i, r_{t-1}^i, r_{t-2}^i, \ldots, r_{t-30}^i, r_{t-60}^i\right]$. Inputs that we call asset states as they directly relate to the portfolio's assets are not only past returns lagged over the δ_1 periods but also the standard deviation. The intuition behind the consumption of returns standard deviation or equivalently their volatility is that volatility is a good predictor of crisis. Indeed it is a stylized fact in the financial literature that volatility is a good predictor of risk Ross [1976], Harmon et al. [2010] and that an increase in volatility comes swiftly after a market crash Black [1976], Wu [2001]. The period to compute the volatility is a hyper parameter, again another hyper-parameter to fine-tune and is arbitrarily taken to 20 periods to represent a month of data. If we summarize, asset states A_t are given by two matrices $A_t = \left[A_t^1, A_t^2\right]$ with the first matrix A_t^1 (in red) being the the matrix of returns:

$$A_t^1 = \begin{pmatrix} r^1(t) & \ldots & r^1(t-60) \\ \ldots & \ldots & \ldots \\ r^m(t) & \ldots & r^m(t-60) \end{pmatrix}$$

and the second matrix (in blue) A^2 being standard deviations:

$$A_t^2 = \begin{pmatrix} \sigma^1(t) & \ldots & \sigma^1(t-60) \\ \ldots & \ldots & \ldots \\ \sigma^m(t) & \ldots & \sigma^m(t-60) \end{pmatrix}$$

The asset states are stored in a 3-D tensor as shown in figure 3.2. Its similarities with an image where pixels are stored in 3 different matrices representing red, green, and blue images enable us

[1] as there are approximately 60 trading days in a quarter and 20 days in a month

to use a 2 dimensional convolution network for our deep network. The analogy goes even further as it is well known in image recognition that convolutional networks achieve strong performances thanks to their capacity to extract meaningful features and to have very limited parameters hence avoiding over-fitting.

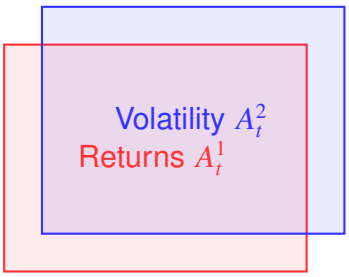


Figure 3.2: 3 dimensional tensor: the asset states

To introduce conceptual based information, the asset manager robot observes also additional important features denoted by C_t that provide insights about the future evolution of the portfolio strategies. Using market knowledge from Homa capital multi assets solutions, we add 3 features (referred to as contextual features) that are correlation between equity and bonds denoted by c_t^1, Citigroup global economic surprise index denoted by c_t^2, and risk aversion index denoted by c_t^3. These features are not taken at random but are well known or at least assumed to have some predictive power for our portfolio strategies as these strategies incorporate a mix of equity and bonds and are highly sensitive to economic surprise and risk aversion level. Again to ensure somehow some Markovianity and to include in the current knowledge of the virtual agent more than the last observation of these features, we introduce a second lag operator δ_2 that operates on the contextual features. To keep things simple in our experience, we take the same vector of lagging periods to represent this second lag operator although the method can be fine-tuned with two different lags for the asset and contextual states. In our setting, $\delta_2 = [0, 1, 2, 3, 4, 20, 60]$ and the contextual states that is represented by C_t writes as follows:

$$C_t = \begin{pmatrix} c^1(t) & \dots & c^1(t-60) \\ \dots & \dots & \dots \\ c^3(t) & \dots & c^3(t-60) \end{pmatrix}$$

In contrast to asset states, contextual states C_t are only represented by a two dimensional tensor or equivalently a matrix. If we want to use convolutional networks, we, therefore, need to use 1D (for 1 dimensional) and not 2D (2 dimensional) convolutions. In addition, we add in these common contextual features the maximum portfolio strategy's return, and the maximum and minimum portfolio strategy's volatilities. The latter two are like asset states motivated by the stylized fact that standard deviations are useful features to detect crisis.

Last but not least we can also introduce that our state s_t incorporates the previous portfolio allocation. Hence our state can take the following three inputs:

- previous portfolio strategy returns lagged by δ_1 called the asset states A_t;
- contextual features observed lagged by δ_2 called the common states C_t;

- the previous weight allocation w_{t-1};

or mathematically, $s_t = \{A_t, C_t, w_{t-1}\}$

Our optimal control problem is to find the optimal policy $\pi^*(s_t)$ that maximizes the total reward denoted by $R(T)$ for one episode. Under very strong theoretical assumptions, this optimal policy always exists and is unique. In practice, we are far from the theoretical framework and we may find only locally optimal policies thanks to gradient ascent! The policy is represented by a deep network whose parameters are given by θ and composed of three sub-networks as illustrated in figure 3.3 and further described in 3.3.1. Hence the optimal control problem writes as

$$\max_{\theta} \mathbf{E}_{\pi_{\theta}(.)} R(T),$$

where $\mathbf{E}_{\pi_\theta}$ represents the expectation under the assumption that our policy $\pi_\theta(s_t)$ is precisely represented by our deep networks whose parameters are θ for a state at time t given by s_t. The total reward $R(T)$ can either be the net performance of the portfolio or some risk return criterion like the Sharpe ratio computed as the ratio of the average mean return over its standard deviation.

3.3.1 Network Architecture

Our network (as described in figure 3.3) uses three types of inputs:

- sub-network 1: portfolio returns and standard deviations observed over the lag δ_1 array (the asset states A_t);
- sub-network 2: contextual information given by the correlation between equities and bonds, the Citigroup economic surprise and the risk aversion indexes observed over the lag δ_2 array and other additional common features like the maximum portfolio strategy's return, the maximum and minimum portfolio strategy's volatilities (the context states C_t);
- and potentially sub-network 3: the previous portfolio allocation w_{t-1}

We concatenate these 3 networks into a final one using two dense layers and a final softmax one to infer the portfolio weights.

Our reward is either the Sharpe ratio or the net value of the final portfolio. In terms of network internal architecture, we can either use convolution layers for sub-network 1 and 2 (convolution 2D and 1D respectively) or LSTM units. We can also do adversarial training by introducing some Gaussian noise in the training to make each iteration slightly different. This helps to have more robust models.

Concerning the train-validation-test split of our data-set, we use the following split: Train data-set is from 01-Jan-2010 to 31-Dec-2015, validation set from 01-Jan-2016 to 31-Dec-2017, while the test data set ranges from 01-Jan-2018 to 31-Mar-2020. Hyper-parameters are tested on the validation sets. We provide the hyper-parameters used in the final run in table 3.4. Results are quite sensitive to the Adam learning rate and the lag 1 and 2 arrays. We tried various solutions and found that taking the last week of observation, the last month and the last quarter was working well and quite intuitive for the lag 1 and 2 arrays. Results of the various trained networks and performance over iterations can be visualized in http://www.aisquareconnect.com/deeprl/ICPRSummary.html

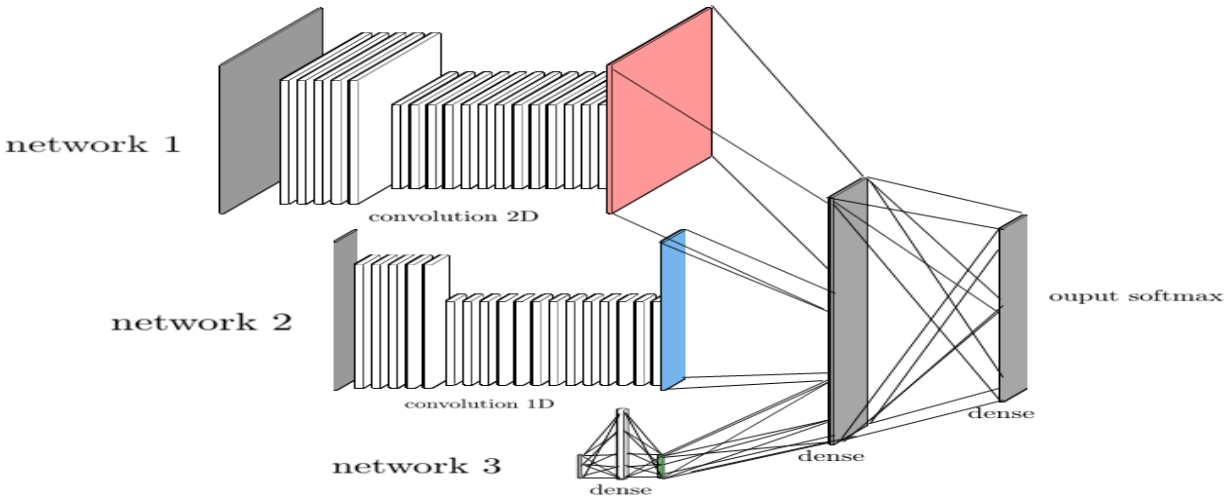


Figure 3.3: Possible DRL network architecture

and are also given as supplementary materials of this chapter.

All in all, the different possible network configurations and architectures represent 32 models whose results are given in table 3.2. In our experiment, $m = 4$ with the first three assets representing real strategies, while the fourth one being just cash whose value does not change over time. To represent the performance of each of the 3 strategies, we plot portfolio 1 which consists of taking only strategy 1 (in blue in figure 3.4), respectively portfolio 2 and 3 taking only strategy 2 and 3 (in orange and green).

It is worth noticing that the portfolio 3 consisting of 100 % in strategy 3 has a strong tendency to over-perform the other two strategies (portfolio 1 and 2). Hence we expect the deep RL agent to allocate mostly in strategy 3 and when anticipating a crisis, to allocate in cash. This is exactly what it does as illustrated in figure 3.5. It is also interesting to notice that the trained deep RL agent is mostly invested in strategy 3 and from time to time swap this allocation to a pure cash allocation. The anticipated crisis in 2018 enables the agent to slightly over-perform portfolio 3 from 2018 to the end of 2019. The agent, however, is not all mighty and makes mistakes as illustrated by the wrong peaked cash allocation in the end of 2019. It is able to adapt to the Covid crisis and to brutally swap allocation from strategy 3 to cash and back as markets bounced back at the end of March.

3.3.2 DRL algorithm

To find the optimal action $\pi^*(s_t)$ (in terms of portfolio allocation), we use deep policy gradient method with non linear activation (Relu). We use buffer replay to memorize all marginal rewards so that we can start batch gradient descent once we reached the final time step. We use the traditional Adam optimization so that we have the benefit of adaptive gradient descent with root mean square propagation Kingma and Ba [2014].

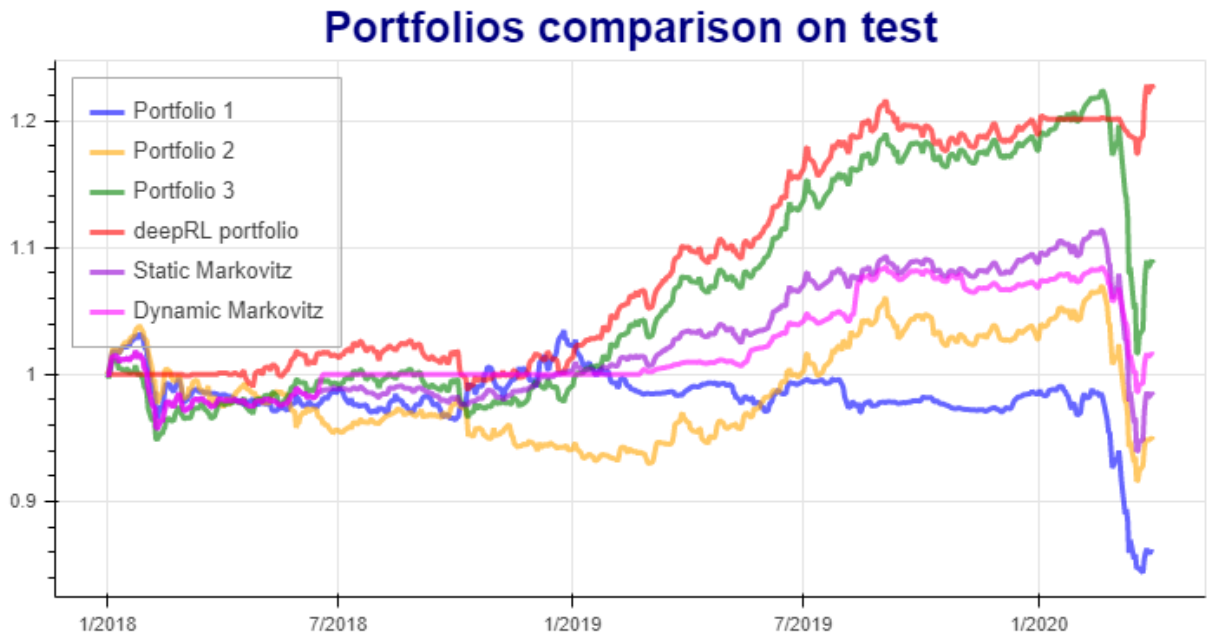


Figure 3.4: Deep RL Portfolio Optimisation Result

3.3.3 Results

Performance results are given below in table 3.2. Best performing models are highlighted in light gray. Returns are computed annually. Hence for a total performance of 21 % (as shown in figure 3.4) over the period of January 1st 2018 to March 31st 2020, the corresponding annual return is 8.8 %. Overall, out of the 32 models available, there are many DRL models that are able to over-perform not only traditional methods like static Markowitz/ but also the best portfolio (sometimes referred to as the naive winner strategy) in terms of net performance and Sharpe ratio, with a final annual net return of 8.8% when using the best net profit reward model or 8.6% when using the best Sharpe ratio reward model compared to 3.9 % for the naive winner method. Dynamic Markowitz/ method consists in computing the Markowitz/ optimal allocation every 3 months. The Naive winner method consists in just selecting the best strategy over the train data set, which is strategy 3.

Table 3.1: Performance results

	Portfolio 1	Portfolio 2	Portfolio 3	Dynamic Markowitz/	Deep RL Net_profit	Deep RL Sharpe	Naive winner
Net Performance	-6.3%	-2.1%	3.9%	0.7%	**8.8%**	8.6%	3.9%
Std dev	6.1%	6.5%	7.3%	4.3%	4.5%	4.2%	7.3%
Sharpe ratio	na	na	0.53	0.17	1.95	**2.08**	0.53

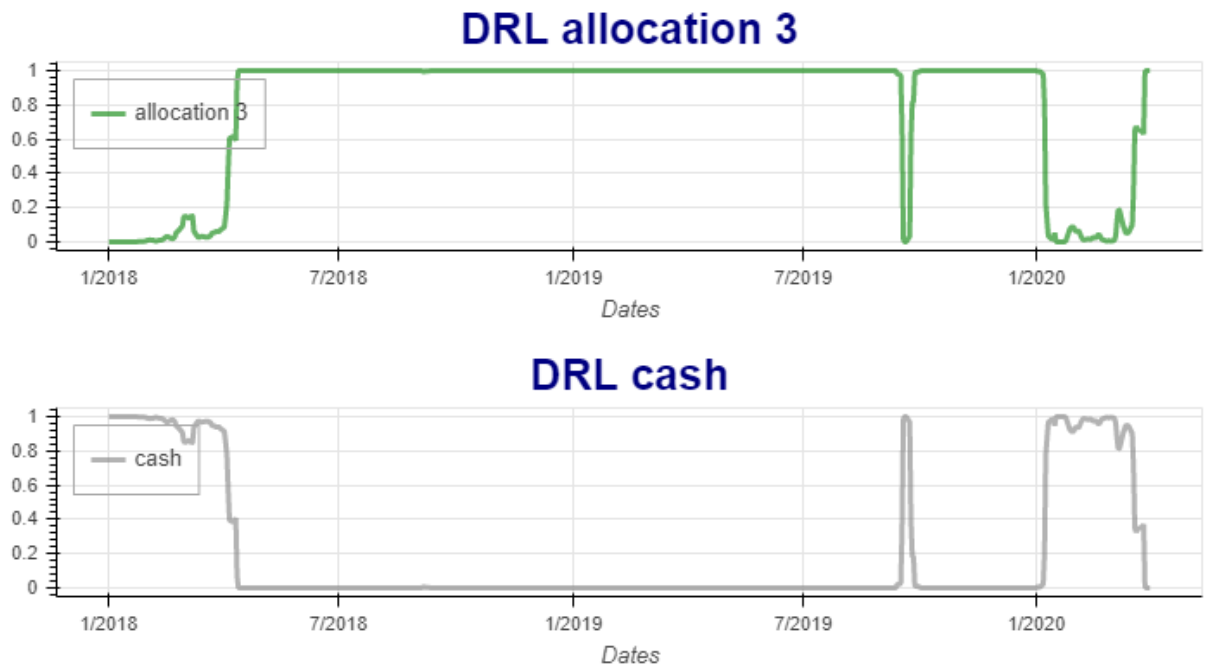


Figure 3.5: Corresponding optimal allocation

3.4 Learning of the network parameters

The agent's objective is to maximize its total reward R given at episode end. This reward can be net portfolio performance or Sharpe ratio computed as portfolio mean return over its standard deviation. Because we somehow play and play again the same scenario with the same reward function, the current framework has two important distinctions from many other RL problems. One is that the domain knowledge of the environment is well-mastered, and can be fully exploited by the agent. This exact expressiveness is a direct consequence that the agent's action has no influence on future price, which is clearly the case for small transactions or liquid assets. This isolation of action and external environment also allows one to use the same segment of market history to evaluate different sequences of actions.

The second distinction is that the final reward depends on all episodic actions. In other words, all episodic actions are important, justifying the full-exploitation approach.

3.4.1 Deterministic Policy Gradient

A policy is a mapping from the state space to the action space, $\pi : \mathcal{S} \to \mathcal{A}$. With full exploitation in the current framework, an action is deterministically produced by the policy of a state. The optimal policy is obtained using a gradient ascent algorithm. To achieve this, a policy is specified by a set of parameters $\vec{\theta}$, and $\vec{a}_t = \pi_{\vec{\theta}}(s_t)$. The performance metric of $\pi_{\vec{\theta}}$ for time interval $[0, t]$ is defined as the corresponding reward function of the interval,

$$J_{[0,t]}(\pi_{\vec{\theta}}) = R\left(\vec{s}_1, \pi_{\vec{\theta}}(s_1), \cdots, \vec{s}_t, \pi_{\vec{\theta}}(s_t), \vec{s}_{t+1}\right). \tag{3.1}$$

After random initialization, the parameters are continuously updated along the gradient direction with a learning rate λ,

$$\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}}). \tag{3.2}$$

To make the gradient ascent optimization, we use the standard Adam (short for Adaptive Moment Estimation) optimizer to have the benefit of adaptive gradient descent with root mean square propagation Kingma and Ba [2014].

3.4.2 Crisis adaptation

It is remarkable that the DRL approach is able to handle the Covid crisis softly as displayed by figure 3.6. If we zoom over the period out of sample from December 2019 to March 2020, we can see that the DRL agent is able to rapidly reduce exposure to strategy 3 and allocate in cash, detecting thanks to contextual information that a crisis is imminent as shown in figure 3.7. Interestingly, the DRL agent reallocates to portfolio 3 in March picking the market rebound.

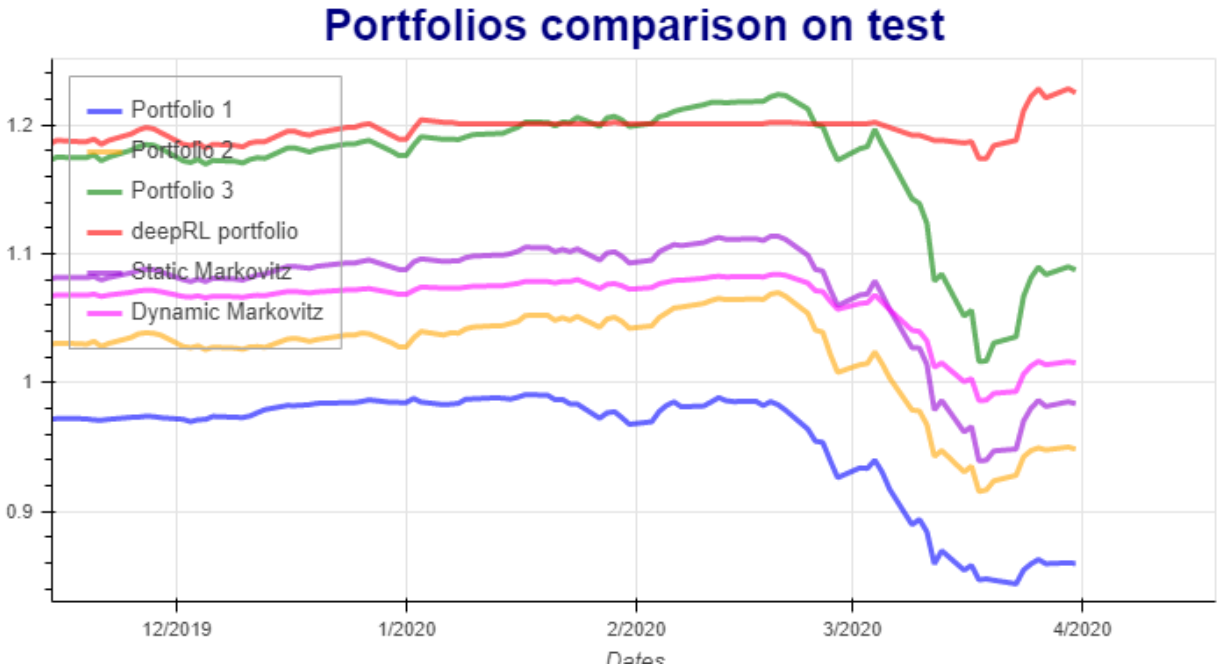


Figure 3.6: Crisis portfolio reaction. It is worth noticing that the best DRL agent has a stable performance as it rapidly dis-invests and put asset in cash during the Covid crisis

Best networks as illustrated in table 3.2 are mostly convolutional networks. We found that for the sub network 1, it is optimal to have 2 convolutional layers but only 1 convolutional layer for the sub network 2. We illustrate the sub-network 1 in figure 3.8.

3.4.3 Impact of contextual information

Logically, networks with contextual information perform better as they have more information. For each network configuration, we compute the difference between the version with and without contextual information. We summarized these results in table 3.3 with the best results highlighted

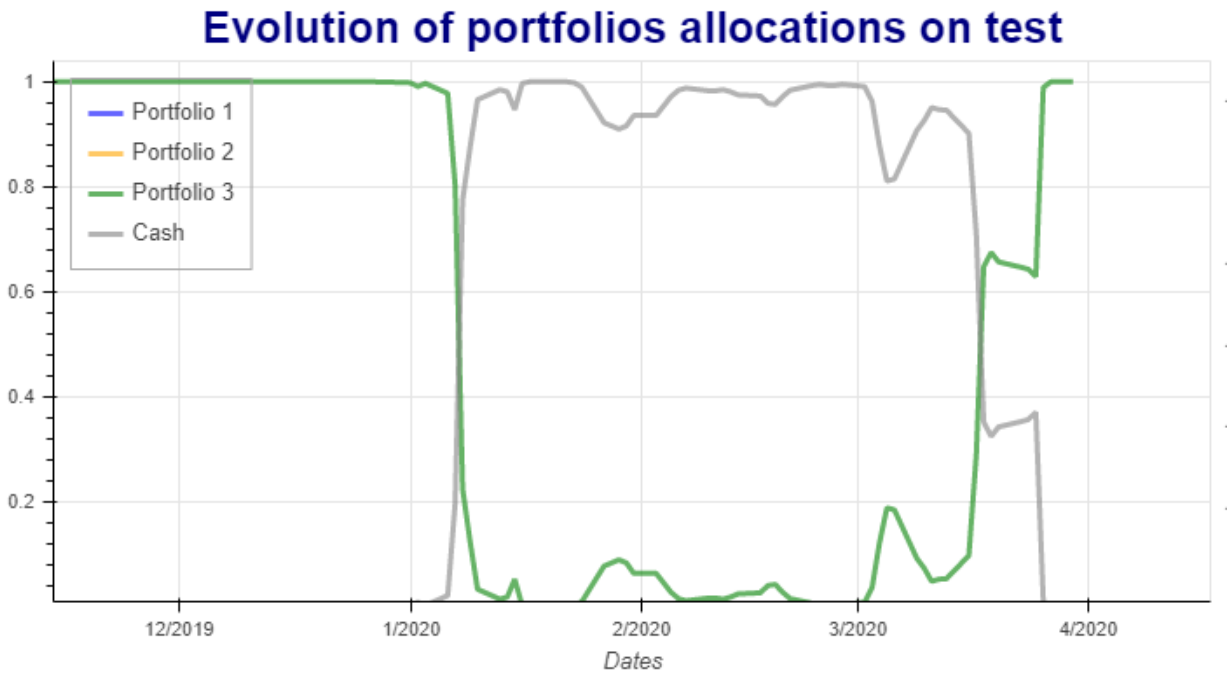


Figure 3.7: Allocation during Covid crisis. It is worth noticing that the best DRL agent allocates almost all assets either in cash or strategy 3 and is rarely mixing the two strategies. It does not include at all strategies 1 or 2, indicating that the optimal choice is to saturate the allocation constraints as we only permit the dynamic agent to allocate between 0 and 100 %

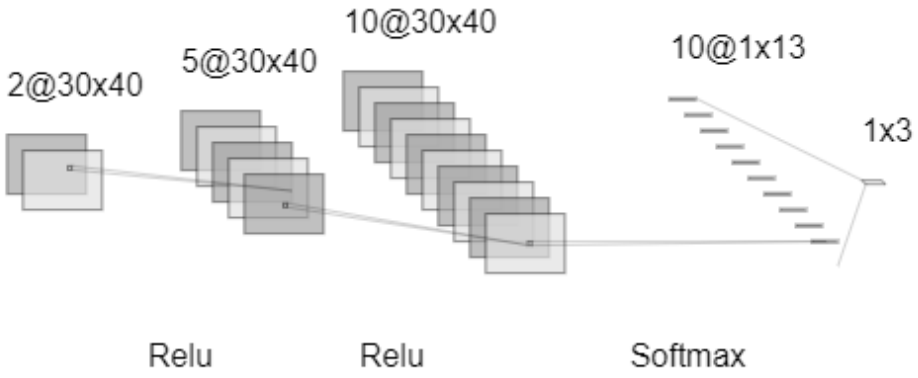


Figure 3.8: convolutional network 1

in yellow. For all configurations, the version with contextual information achieves higher annual returns. This is almost the case also on Sharpe ratio, but there are exceptions. If we remove for each criterion the two largest differences classified as outliers, we found that contextual based models increase on average annual returns by 2.45 % and Sharpe ratio by 0.29.

3.5 Further work

In experiment, we see that the contextual based approach over-performs baseline methods like Markowitz. We also experienced that CNN architecture performs much better than LTSM as it reduces the number of parameters to train and share parameters across portfolio strategies. Adversarial training also makes the training more robust by providing a more challenging environment. Last but not least, it is quite important to fine-tune the numerous hyper-parameters of the contextual based DRL model, namely the various lags (lags period for the sub network fed by portfolio strategies past returns, lags period for common contextual features referred to as the common features), standard deviation period, learning rate, etc... It is compelling that the suggested framework is linearly scalable with the portfolio size and can accommodate contextual information. Our findings suggest that modeling the state with previous weight allocation deteriorates training and does not help suggest that the artifact of introducing previous weight to have a direct impact on state when performing an action is artificial and that in reality under the assumption of small market impact, it is more efficient to assume that portfolio allocation does not influence future state. The memory mechanism is quite beneficial as it allows one to compute the final reward on each episode and hence allows avoiding gradient vanishing problems faced by many deep networks. Moreover, thanks to this memory mechanism, it is not challenging to create an online learning mechanism that can continuously digest incoming market information to improve the dynamic agent. The profitability of this framework surpasses traditional portfolio-selection methods by a non negligible factor as it outperforms dynamic Markowitz by 8 % and the best strategy by 4 %.

This better performance should be mitigated by the fact that the dynamic DRL agent is able to adapt well to the Covid crisis. Hence it benefits from an exceptional and almost unique condition in the financial history. Consequently, these numbers should not be taken literally but rather as a sign of the capacity of deep RL methods to achieve human performance in portfolio allocation and to be able to detect and adapt to crisis patterns. Despite the efficiency of contextual based DRL models in experiments, these models can be improved in future works. Their main weakness is the number of hyper parameters that need to be estimated on the validation set. Their second major weakness relies on the fact that in finance, each experience is somehow unique and one may not be able to draw conclusions on a single test set. Drawing a general conclusion is premature and beyond reason at this stage. It should be tested on more financial markets and on more outcomes. It may also be tested in terms of stability and capacity to adapt to further crisis patterns.

Table 3.2: Results of the various models

Reward	Adversarial training?	Network	Previous weight?	Context?	Annual return	Sharpe
NetProfit	No	Conv2D	No	Yes	**8.8%**	1.95
Sharpe	Yes	Conv2D	No	Yes	8.6%	**2.08**
NetProfit	No	Conv2D	Yes	Yes	8.5%	2.03
Sharpe	No	Conv2D	No	Yes	8.4%	2.01
NetProfit	Yes	Conv2D	No	Yes	8.0%	1.35
NetProfit	Yes	Conv2D	No	No	7.7%	1.94
Sharpe	No	Conv2D	No	No	6.4%	1.31
NetProfit	No	LSTM	No	Yes	6.2%	1.49
NetProfit	No	Conv2D	No	No	5.4%	0.97
Sharpe	Yes	LSTM	No	Yes	5.4%	1.23
NetProfit	Yes	LSTM	No	Yes	5.1%	0.93
NetProfit	Yes	Conv2D	Yes	Yes	4.3%	0.63
Sharpe	Yes	Conv2D	No	No	4.2%	0.69
NetProfit	No	LSTM	Yes	Yes	3.8%	0.52
Sharpe	No	Conv2D	Yes	No	3.8%	0.52
NetProfit	No	Conv2D	Yes	Yes	3.8%	0.52
Sharpe	Yes	LSTM	Yes	Yes	3.8%	0.52
Sharpe	Yes	Conv2D	Yes	Yes	3.7%	0.51
NetProfit	Yes	Conv2D	Yes	No	3.7%	0.51
NetProfit	No	LSTM	Yes	No	3.6%	0.49
NetProfit	Yes	LSTM	Yes	Yes	3.5%	0.48
NetProfit	Yes	LSTM	No	No	3.4%	1.24
Sharpe	No	LSTM	Yes	Yes	3.4%	0.48
NetProfit	No	LSTM	No	No	3.4%	0.47
Sharpe	Yes	Conv2D	Yes	No	3.4%	0.51
NetProfit	Yes	LSTM	Yes	No	2.3%	0.97
Sharpe	Yes	LSTM	No	No	2.3%	0.32
Sharpe	No	LSTM	No	Yes	1.5%	0.22
Sharpe	No	Conv2D	Yes	No	0.9%	0.13
Sharpe	Yes	LSTM	Yes	No	-5.1%	na
Sharpe	No	LSTM	No	No	-5.1%	na
Sharpe	No	LSTM	Yes	No	-5.1%	na

Table 3.3: Difference in returns and sharpe between model with and without contextual information

Reward	Adversarial training?	Network	Previous weight?	Annual return difference	Sharpe difference
NetProfit	No	Conv2D	No	3.43%	0.99
NetProfit	No	Conv2D	Yes	4.68%	**1.51**
NetProfit	No	LSTM	No	2.77%	1.03
NetProfit	No	LSTM	Yes	0.27%	0.03
NetProfit	Yes	Conv2D	No	0.32%	- 0.59
NetProfit	Yes	Conv2D	Yes	0.58%	0.12
NetProfit	Yes	LSTM	No	1.70%	- 0.32
NetProfit	Yes	LSTM	Yes	1.16%	- 0.48
Sharpe	No	Conv2D	No	2.02%	0.70
Sharpe	No	Conv2D	Yes	2.94%	0.40
Sharpe	No	LSTM	No	6.56%	0.22
Sharpe	No	LSTM	Yes	8.52%	0.48
Sharpe	Yes	Conv2D	No	4.39%	1.39
Sharpe	Yes	Conv2D	Yes	0.36%	0.01
Sharpe	Yes	LSTM	No	3.05%	0.91
Sharpe	Yes	LSTM	Yes	**8.87%**	0.52

Table 3.4: Hyper parameters used

hyper-parameters	value	description
batch size	50	Size of mini-batch during training
regularization coefficient	1e-8	L_2 regularization coefficient applied to network training
learning rate	0.01	Step size parameter in Adam
standard deviation period	20 days	period for standard deviation in asset states
commission	10 bps	commission rate
stride	2,1	stride used in convolution networks
conv number 1	5,10	number of convolutions in sub-network 1
conv number 2	2	number of convolutions in sub-network 2
lag period 1	$[60, 20, 4, 3, 2, 1, 0]$	lag period for asset states
lag period 2	$[60, 20, 4, 3, 2, 1, 0]$	lag period for contextual states
noise	0.002	adversarial Gaussian standard deviation

3.6 Summary

In this initial chapter of the model-free part, we address the challenging task of detecting and adapting portfolio allocation to crisis environment. Our approach is based on deep reinforcement learning using contextual information thanks to a second sub-network. The model takes not only past performances of portfolio strategies over different rolling periods, but also portfolio strategies standard deviation as well as contextual information like risk aversion, Citigroup economic surprise index, correlation between equity and bonds over a rolling period to make the best allocation decision. The additional contextual information makes the learning of the dynamic asset manager agent more robust to crisis environments as the agent reacts more rapidly to changing environments. In addition, the usage of standard deviation of portfolio strategies provides a good hint for future crisis. The model achieves better performance than standard financial models. There is room for further improvement as this model constitutes only a first attempt to find a reasonable DRL solution to adapt to crisis situation and to answer positively if DRL can reach a human level in applications to financial problems and in particular in detecting pattern crisis.

This chapter using deep reinforcement learning has been the subject of various publications: Benhamou et al. [2021a], Benhamou et al. [2021e], Benhamou et al. [2020b], Saltiel et al. [2020]. In the next chapter, we will show that DRL generalizes indeed the traditional portfolio methods.

4 Beyond markowitz with deep reinforcement learning

4.1 Motivations

While researchers in the asset management industry have mostly focused on techniques based on financial and risk planning techniques like Markowitz efficient frontier, minimum variance, maximum diversification, or equal risk parity, in parallel, another community in machine learning has started working on reinforcement learning and more particularly deep reinforcement learning to solve other decision-making problems for challenging task like autonomous driving, robot learning, and on more conceptual side games solving like Go. This chapter aims to bridge the gap between these two approaches by showing Deep Reinforcement Learning (DRL) techniques can shed new light on portfolio allocation thanks to a more general optimization setting that casts portfolio allocation as an optimal control problem that is not just a one-step optimization, but rather a continuous control optimization with a delayed reward. The advantages are numerous: (i) DRL maps directly market conditions to actions by design and hence should adapt to changing environments, (ii) DRL does not rely on any traditional financial risk assumptions like that risk is represented by variance, (iii) DRL can incorporate additional data and be a multi inputs method as opposed to more traditional optimization methods. We present an experiment with some encouraging results using convolution networks.

4.2 Introduction

In asset management, there is a gap between mainstream used methods and new machine learning techniques around reinforcement learning and in particular deep reinforcement learning. The former methods rely on financial risk optimization and solve the planning problem of the optimal portfolio as a single-step optimization question. The latter does not make any assumptions about risk, does a more involving multi-steps optimization, and solves complex and challenging tasks like autonomous driving Wang et al. [2018], learning advanced locomotion and manipulation skills from raw sensory inputs Levine et al. [2015, 2016], Schulman et al. [2015a, 2017], Lillicrap et al. [2015] or on a more conceptual side for reaching supra human level in popular games like Atari Mnih et al. [2013], Go Silver et al. [2016, 2017], StarCraft II Vinyals et al. [2019], etc ...

One of the reasons often put forward for this situation is that asset management researchers have mostly been trained with an econometric and financial mathematics background, while the deep reinforcement learning community has been mostly trained in computer science and robotics, leading to two distinctive research communities that do not interact much between each other. In this chapter, we aim to present the various approaches to show similarities and differences to bridge the gap between these two approaches. Both methods can help solve the decision-making problem of finding the optimal portfolio allocation weights.

4.2.1 Related works

As this chapter aims at bridging the gap between traditional asset management portfolio selection methods and deep reinforcement learning, there are too many works to be cited.

On the traditional methods side, the seminal work is Markowitz [1952b] that has led to various extensions like minimum variance Chopra and Ziemba [1993], Haugen and Baker [1991], Kritzman [2014], maximum diversification Choueifaty and Coignard [2008], Choueifaty et al. [2012], maximum decorrelation Christoffersen et al. [2010], risk parity Maillard et al. [2010], Roncalli and Weisang [2016]. We will review these works in the section entitled *Traditional methods*.

On the reinforcement learning side, the seminal book is Sutton and Barto [2018]. The field of deep reinforcement learning is growing every day at an unprecedented pace, making the citation exercise complicated. But in terms of breakthroughs of deep reinforcement learning, one can cite the workaround Atari games from raw pixel inputs Mnih et al. [2013, 2015], Go Silver et al. [2016, 2017], StarCraft II Vinyals et al. [2019], learning advanced locomotion and manipulation skills from raw sensory inputs Levine et al. [2015, 2016] Schulman et al. [2015a,b, 2017], Lillicrap et al. [2015], autonomous driving Wang et al. [2018] and robot learning Gu et al. [2017].

On the application of deep reinforcement learning methods to portfolio allocations, there is already a growing interest as recent breakthroughs have put a growing emphasis on this method. Hence, the field is growing very rapidly and surveys like Fischer [2018] are already outdated. Driven initially mostly by applications to cryptocurrencies and Chinese financial markets Jiang and Liang [2016], Zhengyao et al. [2017], Liang et al. [2018], Yu et al. [2019], Wang and Zhou [2019], Benhamou et al. [2021a, 2020b], the field is progressively taking off on other assets Kolm and Ritter [2019], Liu et al. [2020], Ye et al. [2020], Li et al. [2019], Xiong et al. [2019]. More generally, DRL has recently been applied to other problems than portfolio allocation. For instance, Deng et al. [2016], Zhang et al. [2019], Huang [2018], Théate and Ernst [2020], Chakraborty [2019], Nan et al. [2020], Wu et al. [2020] tackle the problem of direct trading strategies Bao and yang Liu [2019] handles the one of multi agent trading while Ning et al. [2018] examine optimal execution.

4.3 Traditional methods

We are interested in finding an optimal portfolio that makes the planning problem quite different from the standard planning problem where the aim is to plan a succession of tasks. Typical planning algorithms are variations around STRIPS Fikes and Nilsson [1971], that start by analyzing ending goals and means, build the corresponding graph, and find the optimal graph. Indeed we start from

the goals to achieve and try to find means that can lead to them. New work like Graphplan as presented in Blum and Furst [1995] uses a novel planning graph, to reduce the amount of search needed, while hierarchical task network (HTN) planning leverages the classification to structure networks and hence reduce the number of graph searches. Other algorithms like search algorithms as A^*, B^*, weighted A^* or for full graph search, branch and bound and its extensions, as well as evolutionary algorithms like particle swarm, CMA-ES are also used widely in AI planning etc.. However, when it comes to portfolio allocation, standard methods used by practitioners rely on more traditional financial risk-reward optimization problems and follow the Markowitz approach.

4.3.1 Markowitz

The intuition of Markowitz portfolio is to be able to compare various assets and assemble them taking into account both return and risk. Comparing just returns of some financial assets would be too naive. One has to take into account her/his investment decision returns with associated risk. Risk is not an easy concept. In Modern Portfolio Theory (MPT), the risk is represented by the variance of the asset returns. If we take various financial assets and display their returns and risk as in figure 4.1, we can find an efficient frontier. Indeed there exists an efficient frontier represented by the red dot line.

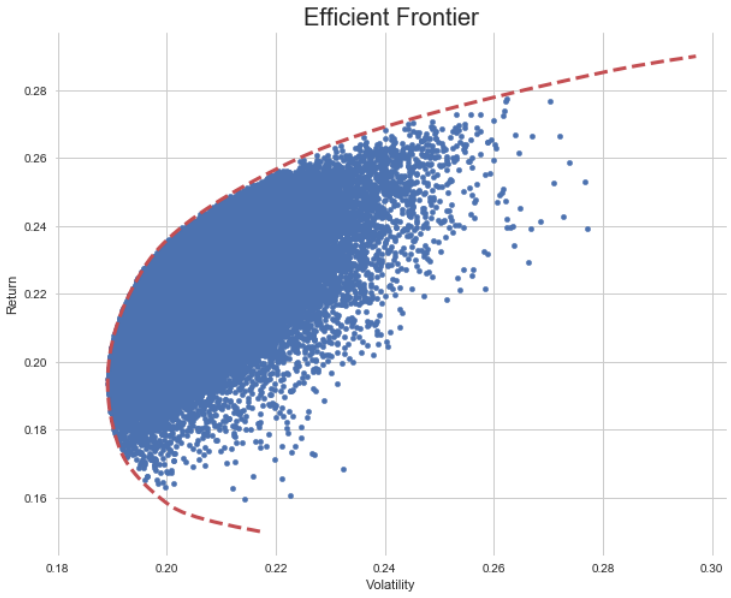


Figure 4.1: Markowitz efficient frontier for the GAFA: returns taken from 2017 to end of 2019

Mathematically, if we denote by $w = (w_1, ..., w_l)$ the allocation weights with $1 \geq w_i \geq 0$ for $i = 0...l$, summarized by $1 \geq w \geq 0$, with the additional constraints that these weights sum to 1: $\sum_{i=1}^{l} w_i = 1$, we can see this portfolio allocation question as an optimization.
Let $\mu = (\mu_1, ..., \mu_l)^T$ be the expected returns for our l strategies and Σ the matrix of variance covariances of the l strategies' returns. Let r_{min} be the minimum expected return. The Markowitz optimization problem to solve is to minimize the risk given a target of minimum expected return as follows:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & w^T \Sigma w \\ \text{subject to} \quad & \mu^T w \geq r_{min}, \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4.1}$$

It is solved by standard quadratic programming. Thanks to duality, there is an equivalent maximization with a given maximum risk σ_{max} for which the problem writes as follows:

$$\begin{aligned} \underset{w}{\text{Maximize}} \quad & \mu^T w \\ \text{subject to} \quad & w^T \Sigma w \leq \sigma_{max}, \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4.2}$$

4.3.2 Minimum variance portfolio

This seminal model has led to numerous extensions where the overall idea is to use a different optimization objective. As presented in Chopra and Ziemba [1993], Haugen and Baker [1991], Kritzman [2014], we can for instance be interested in just minimizing risk (as we are not so much interested in expected returns), which leads to the minimum variance portfolio given by the following optimization program:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & w^T \Sigma w \\ \text{subject to} \quad & \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4.3}$$

Maximum diversification portfolio

Denoting by σ the volatilities of our l strategies, whose values are the diagonal elements of the covariance matrix Σ: $\sigma = (\Sigma_{i,i})_{i=1..l}$, we can shoot for maximum diversification with the diversification of a portfolio defined as follows: $D = \frac{w^T \sigma}{\sqrt{w^T \sum w}}$. We then solve the following optimization program as presented in Choueifaty and Coignard [2008], Choueifaty et al. [2012]

$$\begin{aligned} \underset{w}{\text{Maximize}} \quad & \frac{w^T \sigma}{\sqrt{w^T \sum w}} \\ \text{subject to} \quad & \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4.4}$$

The concept of diversification is simply the ratio of the weighted average of volatilities divided by the portfolio volatility.

Maximum decorrelation portfolio

Following Christoffersen et al. [2010] and denoting by C the correlation matrix of the portfolio strategies, the maximum decorrelation portfolio is obtained by finding the weights that provide the maximum decorrelation or equivalently the minimum correlation as follows:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & w^T C w \\ \text{subject to} \quad & \sum_{i=1\ldots l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4.5}$$

Risk parity portfolio

Another approach following risk parity Maillard et al. [2010], Roncalli and Weisang [2016] is to aim for more parity in risk and solve the following optimization program

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & \frac{1}{2} w^T \Sigma w - \frac{1}{n} \sum_{i=1}^{l} \ln(w_i) \\ \text{subject to} \quad & \sum_{i=1\ldots l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4.6}$$

All these optimization techniques are the usual way to solve the planning question of getting the best portfolio allocation. We will see in the following section that there are many alternatives leveraging machine learning that removes cognitive bias of risk and are somehow more able to adapt to changing environments.

4.4 Unsupervised learning

It is interesting that the above methods (Minimum variance, maximum diversification, risk parity, etc) have some flavor of unsupervised learning or representation learning tasks. In a sense, they aim to group assets into clusters based on their profitability and risk and allocate more funds to the most efficient ones. Obviously, we could also use other unsupervised learning methods to create corresponding portfolios.

4.4.1 PCA and eigen portfolios

The most obvious algorithm to do unsupervised learning and be able to classify data into a set of reduced dimensions and orthogonal (linearly uncorrelated) vectors is principal component analysis (PCA). PCA projects data into a much smaller dimensional space by maximizing the variance of each dimension, represented by eigen vectors. Hence the first eigenvector explains most of the data variation represented by its variance-covariance matrix, and the following eigen vectors are sorted in terms of their variance while being orthogonal to previous ones. We could use these variance ranking ideas to create a portfolio that is mostly invested in the first principal component and has decreasing weights across other eigenvectors as in Guo et al. [2018].

4.4.2 Autoencoder risk

The major drawback of PCA is its linear dependence on the data axis. If we want to do non-linear dimensionality reduction, the mainstream method in machine learning is to use autoencoders as depicted in figure 4.2. Autoencoders aim at learning the internal nature of data into a much lower dimensional vector that is able to restore inputs from this representation. Autoencoders can be used in many ways for portfolio selection. Autoencoders can help learning risk carried by the particular asset by focusing on the specific capacity of a given asset to be represented well by the autoencoder network. Hence if some asset movement can't be restored well, meaning its output value from the autoencoder differs a lot from its input with the difference measured by its mean squared error, this asset is assumed to carry more risk. We can therefore classify assets in terms of learned risk thanks to autoencoder as explained in Heaton et al. [2016].

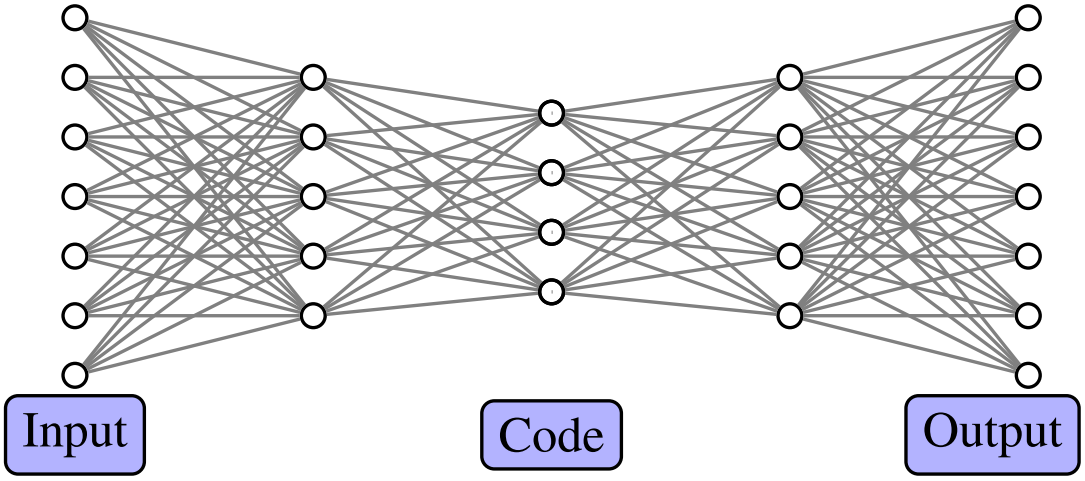


Figure 4.2: Autoencoder

4.4.3 Hierarchical Risk Parity

Geometrically, a covariance matrix of the assets in the portfolio is a complete graph (on the left), can we figure out a tree-based model that will be more optimal? One of the optimization-based portfolio management methods is a risk parity model. It is also stated as an optimization problem, where we allocate rather the risk than the capital resources. The problem of this approach (and, actually, most of the approaches described in this chapter) matures when we are working with portfolios of very huge size if we represent connections between assets geometrically, they will be in the form of the complete graph (see image above), which is an over-complication in the world of hierarchies. The solution lies again in unsupervised learning, but with the exploitation of the hierarchical clustering algorithms applied to the covariance matrix. After finding clusters of the assets, we can re-allocate risk over them recursively.

4.5 Reinforcement learning

Previous financial methods treat the portfolio allocation planning question as a one-step optimization problem, with convex objective functions. There are multiple limitations to this approach:

- they do not relate market conditions to portfolio allocation dynamically.
- they do not take into account that the result of the portfolio allocation may potentially be evaluated much later.
- they make strong assumptions about risk.

What if we could cast this portfolio allocation planning question as a dynamic control problem where we have some market information and need to decide at each time step the optimal portfolio allocation problem and evaluate the result with delayed reward? What if we could move from static portfolio allocation to optimal control territory where we can change our portfolio allocation dynamically when market conditions change. Because the community of portfolio allocation is quite different from the one of reinforcement learning, this approach has been ignored for quite some time even though there is a growing interest in the use of reinforcement learning and deep reinforcement learning over the last few years. We will present here in greater detail what deep reinforcement is in order to suggest more discussions and exchanges between these two communities.

Contrary to supervised learning, reinforcement learning does not try to predict future returns. It does not either try to learn the structure of the market implicitly. Reinforcement learning does more: it directly learns the optimal policy for the portfolio allocation in connection with the dynamically changing market conditions.

4.5.1 Deep Reinforcement Learning Intuition

As its name stands for, Deep Reinforcement Learning (DRL) is the combination of Reinforcement Learning (RL) and Deep (D). The use of deep learning is to represent the policy function in RL. In a nutshell, the setting for applying RL to portfolio management can be summarized as follows:

- current knowledge of the financial markets is formalized via a state variable denoted by s_t.
- Our planning task which is to find an optimal portfolio allocation can be thought of as taking an action a_t on this market. This action is precisely the decision of the current portfolio allocation (also called portfolio weights).
- Once we have decided on the portfolio allocation, we observe the next state s_{t+1}.
- We use a reward to evaluate the performance of our actions. In our particular setting, we can compute this reward only at the final time of our episode, making it quite special compared to a standard reinforcement learning problem. We denote this reward by R_T where T is the final time of our episode. This reward R_T is in a sense similar to our objective function in traditional methods. A typical reward is the final portfolio net performance. It could obviously be other financial performance evaluation criteria like Sharpe, Sortino ratio, etc..

Following standard RL, we model our problem to solve with a Markov Decision Process (MDP) as in Sutton and Barto [2018]. MDP assumes that the agent knows all the states of the environment and has all the information to make the optimal decision in every state. The Markov property implies in addition that knowing the current state is sufficient. MDP assumes a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the state action to next state transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and $\mathcal{R}$ is the immediate reward. The goal of the

agent is to learn a policy that maps states to the optimal action $\pi : \mathcal{S} \rightarrow \mathcal{A}$ and that maximizes the expected discounted reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$.

The concept of using deep networks is to represent the function that dynamically relates the states to the action called in RL the policy and denoted by $\vec{a}_t = \pi(s_t)$. This function is represented by a deep network because of the universal approximation theorem that states that any function can be represented by a deep network provided we have enough layers and nodes. Compared to traditional methods that only solve a one step optimization, we are solving the following dynamic control optimization program:

$$\begin{aligned} \underset{\pi(.)}{\text{Maximize}} \quad & \mathbb{E}[R_T] \\ \text{subject to} \quad & a_t = \pi(s_t) \end{aligned} \tag{4.7}$$

Note that we maximize the expected value of the cumulated reward $\mathbb{E}[R_T]$ because we are operating in a stochastic environment. To make things simpler, let us assume that the cumulated reward is the final portfolio net performance. Let us write P_t the price at time t of our portfolio, and its return at time t: r_t^P and the portfolio assets return vector at time t: $\vec{r}_t$. The final net performance is written as $P_T/P_0 - 1 = \prod_{t=1}^{T}(1 + r_t^P) - 1$. The returns r_t^P is a function of our planning action a_t as follows: $(1 + r_t^P) = 1 + \langle \vec{a}_t, \vec{r}_t \rangle$ where $\langle \cdot, \cdot \rangle$ is the standard inner product of two vectors. In addition, if we recall that the policy is parametrized by some deep network parameters, θ: $a_t = \pi_\theta(s_t)$, we can make our optimization problem slightly more detailed as follows:

$$\begin{aligned} \underset{\theta}{\text{Maximize}} \quad & \mathbb{E}\left[\prod_{t=1}^{T}(1 + \langle \vec{a}_t, \vec{r}_t \rangle)\right] \\ \text{subject to} \quad & a_t = \pi_\theta(s_t). \end{aligned} \tag{4.8}$$

It is worth noticing that compared to previous traditional planning methods (optimization 4.1, 4.3, 4.4, 4.5 or 4.5), the underlying optimization problem in RL 4.7 and its rewriting in terms of deep network parameters θ as presented in 4.8 have many differences:

- First, we are trying to optimize a function π and not simple weights w_i. Although this function at the end is represented by a deep neural network that has admitted also weights, this is conceptually very different as we are optimizing in the space of functions $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that is a much bigger space than simply $\mathbb{R}^l$.
- Second, it is a multi time step optimization as it involves results from time $t = 1$ to $t = T$, making it also more involving.

4.5.2 Partially Observable Markov Decision Process

If there is in addition some noise in our data and we are not able to observe the full state, it is better to use Partially Observable Markov Decision Process (POMDP) as presented initially in Astrom [1969]. In POMDP, only a subset of the information of a given state is available. The

partially-informed agent cannot behave optimally. He uses a window of past observations to replace states as in a traditional MDP.

Mathematically, POMDP is a generalization of MDP. POMPD adds two more variables in the tuple, $\mathcal{O}$ and $\mathcal{Z}$ where $\mathcal{O}$ is the set of observations and $\mathcal{Z}$ is the observation transition function $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$. At each time, the agent is asked to take an action $a_t \in \mathcal{A}$ in a particular environment state $s_t \in \mathcal{S}$, that is followed by the next state s_{t+1} with $\mathcal{P}(s_{t+1}|s_t, a_t)$. The next state s_{t+1} is not observed by the agent. It rather receives an observation $o_{t+1} \in \mathcal{O}$ on the state s_{t+1} with probability $Z(o_{t+1}|s_{t+1}, a_t)$.

From a practical standpoint, the general RL setting is modified by taking a pseudo state formed with a set of past observations $(o_{t-n}, o_{t-n-1}, \ldots, o_{t-1}, o_t)$. In practice to avoid large dimension and the curse of dimension, it is useful to reduce this set and take only a subset of these past observations with $j < n$ past observations, such that $0 < i_1 < \ldots < i_j$ and $i_k \in \mathbb{N}$ is an integer. The set $\delta_1 = (0, i_1, \ldots, i_j)$ is called the observation lags. In our experiment we typically use lag periods like (0, 1, 2, 3, 4, 20, 60) for daily data, where (0, 1, 2, 3, 4) provides last week's observation, 20 is for the one-month ago observation (as there are approximately 20 business days in a month) and 60 the three-month ago observation.

4.5.3 Observations

Regular observations

There are two types of observations: regular and contextual information. Regular observations are data directly linked to the problem to solve. In the case of an asset management framework, regular observations are past prices observed over a lag period $\delta = (0 < i_1 < \ldots < i_j)$. To normalize data, we rather use past returns computed as $r_t^k = \frac{p_t^k}{p_{t-1}^k} - 1$ where p_t^k is the price at time t of the asset k. To give information about regime changes, our trading agent receives also empirical standard deviation computed over a sliding estimation window denoted by d as follows $\sigma_t^k = \sqrt{\frac{1}{d}\sum_{u=t-d+1}^{t}(r_u - \mu)^2}$, where the empirical mean μ is computed as $\mu = \frac{1}{d}\sum_{u=t-d+1}^{t} r_u$. Hence our regular observations is a three dimensional tensor $A_t = \left[A_t^1, A_t^2\right]$

$$\text{with} \quad A_t^1 = \begin{pmatrix} r_{t-i_j}^1 & \ldots & r_t^1 \\ \ldots & \ldots & \ldots \\ r_{t-i_j}^m & \ldots. & r_t^m \end{pmatrix}, \; A_t^2 = \begin{pmatrix} \sigma_{t-i_j}^1 & \ldots & \sigma_t^1 \\ \ldots & \ldots & \ldots \\ \sigma_{t-i_j}^m & \ldots. & \sigma_t^m \end{pmatrix}$$

This setting with two layers (past returns and past volatilities) is quite different from the one presented in Jiang and Liang [2016], Zhengyao et al. [2017], Liang et al. [2018] that uses different layers representing closing, open high low prices. There are various remarks to be made. First, high low information does not make sense for portfolio strategies that are only evaluated daily, which is the case of all the funds. Secondly, open high low prices tend to be highly correlated creating some noise in the inputs. Third, the concept of volatility is crucial to detect regime change and is surprisingly absent from these works as well as from other works like Yu et al. [2019], Wang and Zhou [2019], Liu et al. [2020], Ye et al. [2020], Li et al. [2019], Xiong et al. [2019].

Context observation

Contextual observations are additional information that provides intuition about the current context. For our asset manager, there is other financial data not directly linked to its portfolio assumed to have some predictive power for portfolio assets. Contextual observations are stored in a 2D matrix denoted by C_t with stacked past p individual contextual observations. Among these observations, we have the maximum and minimum portfolio strategies' return and the maximum portfolio strategies' volatility. The latter information is like regular observations motivated by the stylized fact that standard deviations are useful features to detect crisis. The contextual state writes as $C^t = \begin{pmatrix} c_t^1 & \dots & c_{t-i_k}^1 \\ \dots & \dots & \dots \\ c_t^p & \dots & c_{t-i_k}^p \end{pmatrix}$. The matrix nature of contextual states C_t implies in particular that we will use 1D convolutions should we use convolutional layers. All in all, observations that are augmented observations, written as $O_t = [A_t, C_t]$, with $A_t = [A_t^1, A_t^2]$ that will feed the two sub-networks of our global network.

4.5.4 Action

In our deep reinforcement learning the augmented asset manager agent needs to decide at each period in which hedging strategy it invests. The augmented asset manager can invest in l strategies that can be simple strategies or strategies that are also done by asset management agents. To cope with reality, the agent will only be able to act after one period. This is because asset managers have a one day turn around to change their position. We will see in experiments that this one day turnaround lag makes a big difference in results. As it has access to l potential hedging strategies, the output is a l dimension vector that provides how much it invests in each hedging strategy. For our deep network, this means that the last layer is a softmax layer to ensure that portfolio weights are between 0 and 100% and sum to 1, denoted by $(p_t^1, ..., p_t^l)$. In addition, to include leverage, our deep network has a second output which is the overall leverage that is between 0 and a maximum leverage value (in our experiment 3), denoted by lvg_t. Hence the final allocation is given by $lvg_t \times (p_t^1, ..., p_t^l)$.

4.5.5 Reward

In terms of reward, we are considering the net performance of our portfolio from t_0 to the last train date t_T computed as follows: $\frac{P_{t_T}}{P_{t_0}} - 1$.

4.5.6 Multi inputs and outputs

We display in figure 4.3 the architecture of our network. Because we feed our network with both data from the strategies to select but also contextual information, our network is a multiple inputs network.

Additionally, as we want from these inputs to provide not only percentage in the different hedging strategies (with a softmax activation of a dense layer) but also the overall leverage (with a dense layer with one single output neurons), we also have a multi outputs network. Additional hyperparameters that are used in the network as L2 regularization with a coefficient of 1e-8.

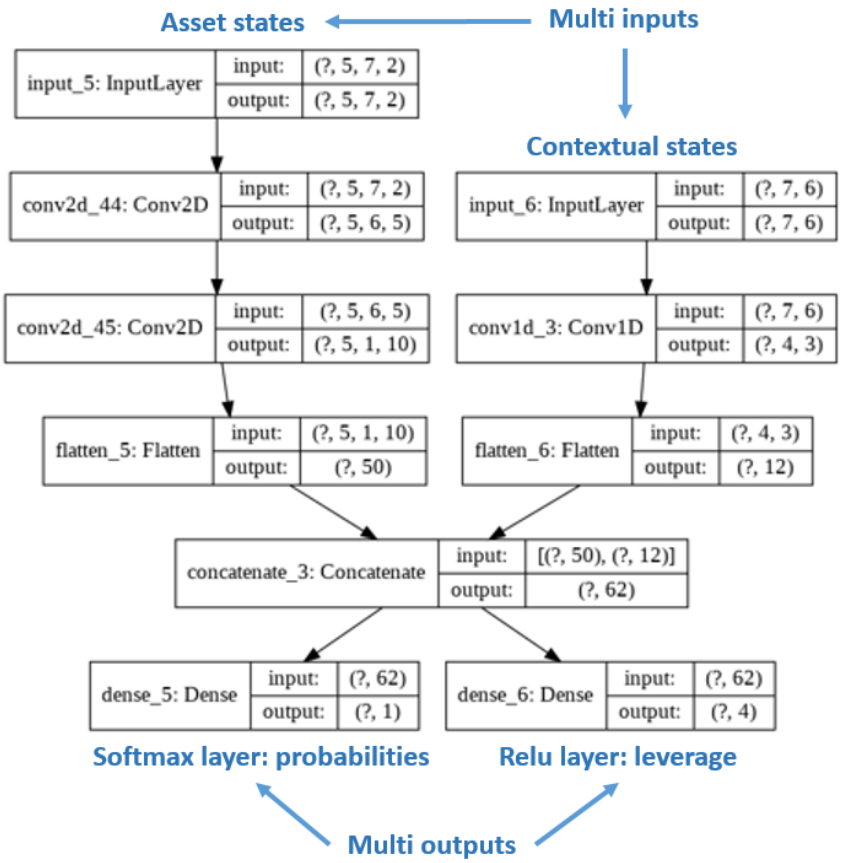


Figure 4.3: network architecture obtained via tensorflow plotmodel function. Our network is very different from standard DRL networks that have single inputs and outputs. Contextual information introduces a second input while the leverage adds a second output

4.5.7 Convolution networks

Because we want to extract some features implicitly with a limited set of parameters, and following Liang et al. [2018], we use a convolution network that performs better than simple fully connected layers. For our so-called *asset states* named like that because there is the part of the states that relate to the asset, we use two layers of convolutional network with 5 and 10 convolutions. These parameters are found to be efficient on our validation set. In contrast, for the contextual states part, we only use one layer of convolution networks with 3 convolutions. We flatten our two sub network in order to concatenate them into a single network.

4.5.8 Adversarial Policy Gradient

To learn the parameters of our network depicted in 4.3, we use a modified policy gradient algorithm called adversarial as we introduce noise in the data as suggested in Liang et al. [2018]. The idea of introducing noise in the data is to have some randomness in each training to make it more robust. This is somehow similar to drop out in deep networks where we randomly perturb the network by randomly removing some neurons to make it more robust and less prone to overfitting. Here, we are perturbing directly the data to create this stochasticity to make the network more robust. A policy is a mapping from the observation space to the action space, $\pi : \mathcal{O} \rightarrow \mathcal{A}$.

To achieve this, a policy is specified by a deep network with a set of parameters $\vec{\theta}$. The action is a vector function of the observation given the parameters: $\vec{a}_t = \pi_{\vec{\theta}}(o_t)$. The performance metric of $\pi_{\vec{\theta}}$ for time interval $[0, t]$ is defined as the corresponding total reward function of the interval $J_{[0,t]}(\pi_{\vec{\theta}}) = R\left(\vec{o}_1, \pi_{\vec{\theta}}(o_1), \cdots, \vec{o}_t, \pi_{\vec{\theta}}(o_t), \vec{o}_{t+1}\right)$.

After random initialization, the parameters are continuously updated along the gradient direction with a learning rate λ: $\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}})$. The gradient ascent optimization is done with standard Adam (short for Adaptive Moment Estimation) optimizer to have the benefit of adaptive gradient descent with root mean square propagation Kingma and Ba [2014]. The whole process is summarized in algorithm 3.

In our gradient ascent, we use a learning rate of 0.01, an adversarial Gaussian noise with a standard deviation of 0.002. We do up to 500 maximum iterations with an early stop condition if, on the train set, there is no improvement over the last 50 iterations.

4.6 Experimental analysis

4.6.1 Description of experimental protocol

We are interested in planning a hedging strategy for a risky asset. The experiment is using daily data from 01/05/2000 to 19/06/2020 for the MSCI and 4 SG-CIB proprietary systematic strategies. The risky asset is the MSCI world index whose daily data can be found on Bloomberg. We choose this index because it is a good proxy for a wide range of asset manager portfolios. The hedging strategies are 4 SG-CIB proprietary systematic strategies further described below. Training and testing are done following extending walk forward analysis as presented in Benhamou et al. [2020c,b, 2021a] with initial training from 2000 to the end of 2006 and testing in a rolling 1 year period. Hence, there are 14 training and testing periods, with the different testing periods

Algorithm 3 Adversarial Policy Gradient

Input: initial policy parameters θ, empty replay buffer $\mathcal{D}$
repeat
  reset replay buffer
  while not terminal **do**
    Observe observation o and select action $a = \pi_\theta(o)$ with probability p and random action with probability $1 - p$,
    Execute a in the environment
    Observe next observation o', reward r, and done signal d to indicate whether o' is terminal
    apply noise to next observation o'
    store (o, a, o') in replay buffer $\mathcal{D}$
    if Terminal **then**
      for however many updates in $\mathcal{D}$ **do**
        compute final reward R
      end for
      update network parameter with Adam gradient ascent $\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}})$
    end if
  end while
until convergence

corresponding to all the years from 2007 to 2020 and training done for a period starting in 2000 and ending one day before the start of the testing period.

4.6.2 Data-set description

Systematic strategies are similar to asset managers that invest in financial markets according to an adaptive, pre-defined trading rule. Here, we use 4 SG CIB proprietary 'hedging strategies', that tend to perform when stock markets are down:

- Directional hedges - react to small negative return in equities,
- Gap risk hedges - perform well in sudden market crashes,
- Proxy hedges - tend to perform in some market configurations, like for example when highly indebted stocks under-perform other stocks,
- Duration hedges - invest in the bond market, a classical diversifier to equity risk in finance.

The underlying financial instruments vary from put options, listed futures, single stocks, to government bonds. Some of those strategies are akin to an insurance contract and bear a negative cost over the long run. The challenge consists in balancing costs versus benefits.

In practice, asset managers have to decide how much of these hedging strategies are needed on top of an existing portfolio to achieve a better risk reward. The decision making process is often based on contextual information, such as the economic and geopolitical environment, the level of risk aversion among investors, and other correlation regimes. The contextual information is modeled by a large range of features :

- the level of risk aversion in financial markets, or market sentiment, measured as an indicator varying between 0 for maximum risk aversion and 1 for maximum risk appetite,
- the bond equity historical correlation, a classical ex-post measure of the diversification benefits of a duration hedge, measured on a 1 month, 3 month and 1 year rolling window,
- The credit spreads of global corporate - investment grade, high yield, in Europe and in the US - known to be an early indicator of potential economic tensions,
- The equity implied volatility, a measure if the 'fear factor' in financial market,
- The spread between the yield of Italian government bonds and the German government bond, a measure of potential tensions in the European Union,
- The US Treasury slope, a classical early indicator for US recession,
- And some more financial variables, often used as a gauge for global trade and activity: the dollar, the level of rates in the US, the estimated earnings per shares (EPS).

A cross-validation step selects the most relevant features. In the present case, the first three features are selected. The rebalancing of strategies in the portfolio comes with transaction costs that can be quite high since hedges use options. Transaction costs are like friction in physical systems. They are taken into account dynamically to penalize solutions with a high turnover rate.

4.6.3 Evaluation metrics

Asset managers use a wide range of metrics to evaluate the success of their investment decision. For a thorough review of those metrics, see for example Cogneau and Hübner [2009]. The metrics we are interested in for our hedging problem are listed below:

- annualized return defined as the average annualized compounded return,
- annualized daily based Sharpe ratio defined as the ratio of the annualized return over the annualized daily based volatility μ/σ,
- Sortino ratio computed as the ratio of the annualized return over the downside standard deviation,
- maximum drawdown (max DD) computed as the maximum of all daily drawdowns. The daily drawdown is computed as the ratio of the difference between the running maximum of the portfolio value defined as $RM_T = \max_{t=0..T}(P_t)$ and the portfolio value over the running maximum of the portfolio value. Hence the drawdown at time T is given by $DD_T = (RM_T - P_T)/RM_T$ while the maximum drawdown $MDD_T = \max_{t=0..T}(DD_t)$. It is the maximum loss in return that an investor will incur if she/he invested at the worst time (at peak).

4.6.4 Results and discussion

Overall, the DRL approach achieves much better results than traditional methods as shown in table 4.1, except for the maximum drawdown (max DD). Because time horizon is important in the comparison we provide risk measures for the last 2 and 5 years to emphasize that the DRL approach seems more robust than traditional portfolio allocation methods.

Table 4.1: Models comparison over 2 and 5 years

	2 Years			
	return	Sortino	Sharpe	max DD
Risky asset	8.27%	0.39	0.36	- 0.34
DRL	**20.64%**	**0.94**	**0.96**	- 0.27
Markowitz	-0.25%	- 0.01	- 0.01	- 0.43
MinVariance	-0.22%	- 0.01	- 0.01	- 0.43
MaxDiversification	0.24%	0.01	0.01	- 0.43
MaxDecorrel	14.42%	0.65	0.63	- 0.21
RiskParity	14.17%	0.73	0.72	**-0.19**
	5 Years			
	return	Sortino	Sharpe	max DD
Risky asset	9.16%	0.57	0.54	- 0.34
DRL	**16.95%**	**1.00**	**1.02**	- 0.27
Markowitz	1.48%	0.07	0.06	- 0.43
MinVariance	1.56%	0.08	0.06	- 0.43
MaxDiversification	1.77%	0.08	0.07	- 0.43
MaxDecorrel	7.65%	0.44	0.39	- 0.21
RiskParity	7.46%	0.48	0.43	**-0.19**

When plotting performance results from 2007 to 2020 as shown in figure 4.4, we see that DRL model is able to deviate upward from the risky asset continuously, indicating a steady performance. In contrast, other financial models are not able to keep their marginal over-performance over time with respect to the risky asset and end slightly below the risky asset.

4.6.5 Allocation chosen by models

The reason for the stronger performance of DRL comes from the way it chooses its allocation. Contrarily to standard financial methods that play the diversification as shown in figure 4.5, DRL aims at choosing a single hedging strategy most of the time and at changing it dynamically, should the financial market conditions change. In a sense, DRL is doing some cherry-picking by selecting what it thinks is the best hedging strategy.

In contrast, traditional models like Markowitz, minimum variance, maximum diversification, maximum decorrelation, and risk parity provide non null weights for all our hedging strategies and do not do cherry-picking at all. They are neither able to change the leverage used in the portfolio as opposed to the DRL model.

4.6.6 Adaptation to the Covid Crisis

The DRL model can change its portfolio allocation should the market conditions change. This is the case from 2018 onwards, with a short deleveraging window emphasized by the small blank disruption during the Covid crisis as shown in figure 4.6. We observe in this figure where we have zoomed over the year 2020, that the DRL model is able to reduce leverage from 300 % to 200 %

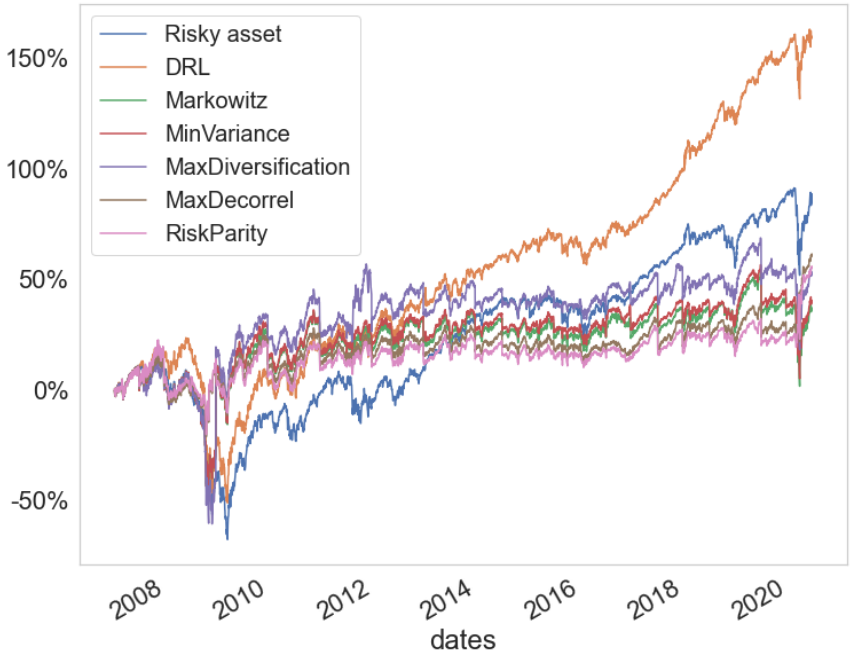


Figure 4.4: We represent the net performance (performance results of a portfolio after the deduction of all fees) in percentage of each model from 2008 to 2020 in order to compare them.

during the Covid crisis (end of February 2020 to the start of April 2020). This is a unique feature of our DRL model compared to traditional financial planning models that do not take leverage into account and keeps a leverage of 300 % regardless of market conditions.

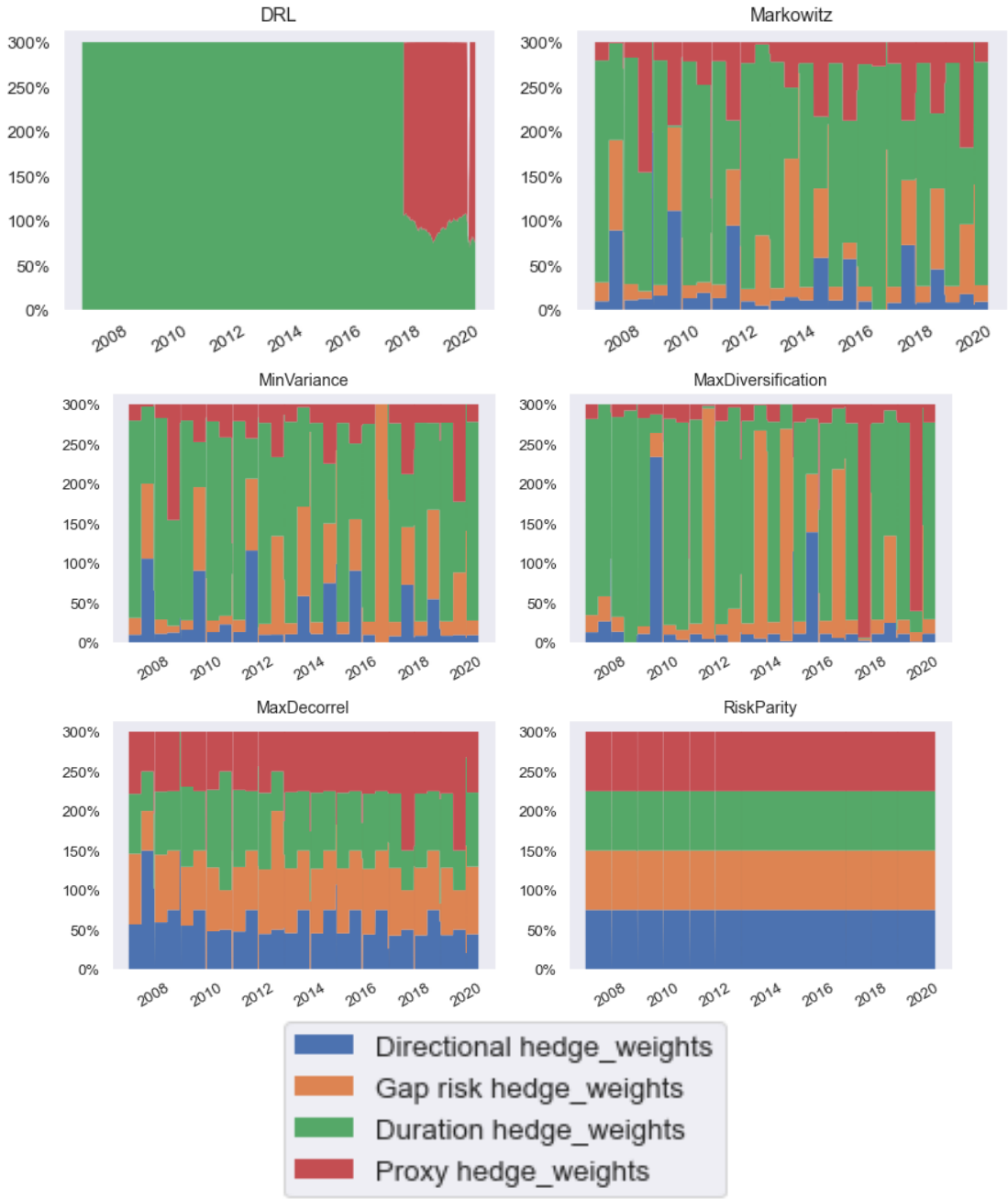


Figure 4.5: Weights for all models: we provide the allocations for the 6 different models (DRL, Markowitz, min variance, maximum diversification, maximum de-correlation, and risk parity). It is worth noting that all traditional models allocate in each of the 4 hedges, namely directional, gap risk, duration, and proxy hedges. In contrast, DRL selects only 2 weights; mostly duration hedges and more recently proxy hedges. This is a particularity of the DRL method that favors concentration in what the model assumes to be the best strategy.

Figure 4.6: Disallocation of DRL model. We focus on the first 6 months of the year 2020 to see that during the Covid crisis in March, the model has been able to reduce allocation the proxy hedge allocation in red while maintaining the duration hedge allocation in green.

4.7 Summary

In this second chapter of the model-free part, we discuss how a traditional portfolio allocation problem can be reformulated as a DRL problem, trying to bridge the gaps between the two approaches. We see that the DRL approach enables us to select fewer strategies, improving the overall results as opposed to traditional methods that are built on the concept of diversification. We also stress that DRL can better adapt to changing market conditions and is able to incorporate more information to make a decision.

As illustrated by the experiment using a similar DRL approach as in the chapter 3, we get improved performance compared to traditional methods. Indeed, the advantages of DRL are numerous: (i) DRL maps directly market conditions to actions by design and hence should adapt to changing environments, (ii) DRL does not rely on any traditional financial risk assumptions, (iii) DRL can incorporate additional data and be a multi inputs method as opposed to more traditional optimization methods. These 2 reasons may intuitively explain why DRL provides improved results to traditional portfolio allocation methods.

As nice as this work is, there is room for improvement as we have only tested a few scenarios and only a limited set of hyper-parameters for our convolutional networks. We should do more intensive testing to confirm that DRL is able to better adapt to changing financial environments than traditional methods and also investigate the impact of more layers and other design choices in our network.

This chapter has been the subject of a publication in a conference Benhamou et al. [2020a]. In the next chapter, we will show that indeed reinforcement learning shares some similarities with supervised learning.

5 Similarities between reinforcement learning and supervised learning

5.1 Motivations

Reinforcement learning (RL) is about sequential decision making and is traditionally opposed to supervised learning (SL) and unsupervised learning (USL). In RL, given the current state, the agent makes a decision that may influence the next state as opposed to SL (and USL) where the next state remains the same, regardless of the decisions taken, either in batch or online learning. Although this difference is fundamental between SL and RL, some connections have been overlooked. In particular, we prove in this chapter that the gradient policy method can be cast as a supervised learning problem where true labels are replaced with discounted rewards. We provide a new proof of policy gradient methods (PGM) that emphasizes the tight link between cross entropy and supervised learning. We provide a simple experiment where we interchange labels and pseudo rewards. We conclude that other relationships with SL could be made if we modify the reward functions wisely.

5.2 Introduction

In Reinforcement Learning (RL), policy gradient methods (PGM) are frequently used Williams [1992], Sutton et al. [1999], Silver et al. [2014], Lillicrap et al. [2015]. PGM are RL techniques that rely upon optimizing the parameters of policies with respect to the expected cumulative reward using gradient descent optimization. They are traditionally opposed to value learning (or value iterations) methods Sutton and Barto [1998], Watkins and Dayan [1992] that either keep improving the value function at each iteration until the value-function converges or optimize the parameters of the value function. The value function is defined as the expected value of the expected cumulative reward conditionally to an initial state and action. PGM principle is very simple. Improve gradually the policy through gradient descent. PGM has two important concepts. They are a policy method as the name emphasizes. They are also a gradient descent method. Policy means we observe and act. Gradient descent methods use the fact that the best move locally is along the gradient. As the move is at first order, a learning rate needs to ensure policy

improvement is not too large at each step. PGM have been popularized in REINFORCE Williams [1992] and in Sutton et al. [1999] and have received wider attention with Actor Critic methods Konda and Tsitsiklis [2003], Peters and Schaal [2008] in particular when using deep PGM Mnih et al. [2016] that combine policy and value methods. In addition, recently, it has been found that an entropy regularization term may fasten convergence O'Donoghue et al. [2016], Nachum et al. [2017], Schulman et al. [2017].

When looking in detail in REINFORCE Williams [1992], we can remark that the gradient term with respect to the policy can indeed be interpreted as the log term in the cross entropy in supervised learning. If in addition, we make the bridge between RL and SL, emphasizing that RL problem can be reformulated as a SL problem where true labels are changed by expected discounted future rewards, and estimated probabilities by policy probabilities, the link between RL and SL becomes obvious. In addition, leveraging the tight relationship between cross entropy and Kullback Leibler divergence, we can interpret the entropy regularization terms very naturally. This is precisely the objective of this short chapter: call attention to the tight connection between RL and SL to give theoretical justifications for some of the techniques used in PGMs.

The chapter is organized as follows. In section 5.4.1, we recall the various choices of functional losses in SL and exhibit that cross entropy is one of the main possibilities for loss functions. We flaunt the relationship between cross entropy and Kullback Leibler divergence, harping on the additional entropy term. In section 5.4.2, we present PGMs. We rub in the interpretation of RL problems as a modified cross entropy SL problem. We conclude in section 5.5 with a financial numerical experience using deep PGM, stressing that in the specific case of actions that do not influence the environment, the difference between RL and SL is very tenuous.

5.3 Related Work

Looking at similarities and synergies between RL and SL together was for a long time overlooked. This can be easily explained as the two research communities were different and thought they were laboring on incompatible or at least very different approaches. However, there has been one type of learning that promotes the similitude between RL and SL. This has been Imitation Learning.

Imitation learning [Schaal, 1996] is a classic technique for learning from human demonstration. Imitation learning uses a supervised approach to imitate an expert's behaviors, hence doing a RL task to accomplish a SL one. DAGGER [Ross et al., 2011] is considered to be the mainstream imitation algorithm. It requests an action from the expert at each step. It uses an action sampled from a mixed distribution from the agent and the expert. It combines the observed states and demonstrated actions to train the agent successively. This has led to numerous extensions of this algorithm and in particular to a deep version of it. Deeply AggreVaTeD [Sun et al., 2017] extends DAGGER with deep neural networks and continuous action spaces. Other approaches have been to stop opposing RL and SL and rather leverage SL in deep RL. Hester et al. [2017] for instance have used SL tasks to train the agent better and created the method of Deep Q-learning from Demonstrations (DQfD). It tackles the problem of the necessity of a huge amount of data for deep reinforcement learning (Deep RL) and the poor performance of Deep RL algorithms during the initial phase of learning. The method of Deep Q-learning from Demonstrations (DQfD) solves the issue by combining RL techniques (temporal difference updates) with SL techniques (supervised classification of the demonstrator's actions) as the target network in Deep Q learning is initially trained with supervised learning.

All of these works show that RL and SL are not as opposed as one may have thought. We argue here that as nice as these works are, they do not emphasize that ignoring for a while the issue of feedback effect of an action on the next state environment, a PGM can be reformulated as a SL task where true labels are changed into future expected reward while the PG can be interpreted as a cross entropy loss minimization.

5.4 Relations between SL and RL

5.4.1 Supervised Learning

SL is quite general and encompasses both SL classification and SL regression. The goal of SL classification is to infer a function from labeled training data that maps inputs into labeled outputs. In psychology, this is sometimes analyzed as concept learning. The deduction of the function parameters is done traditionally through the optimization of a loss function. SL classifiers parameters are the ones of the optimal solution of the optimization program. To keep things simple, let us assume that we are looking at a binary classification problem. Let us assume we observe $D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ that are n independent random copies of $(X, Y) \in \mathcal{X} \times \mathcal{Y}$. The feature X lives in some abstract space $\mathcal{X}$ ($\mathbf{R}^d$ for instance) and Y is called a label. Binary classification assumes that Y take two different values: $\mathcal{Y} = \{-1, 1\}$, while multi-class classification assumes $\mathcal{Y} = \{1, \dots, K\}$. To keep things simple, we will only look at binary classification. Naturally, one would like to find a function $f : \mathcal{X} \mapsto \mathbf{R}$ that best maps X to Y. We are also given a loss function $\ell : \{-1, 1\} \times \{-1, 1\} \mapsto \mathbf{R}$ that measures the error of a specific prediction. The loss function value at an arbitrary point $(Y, \hat{Y})$ reads as the cost incurred when predicting $\hat{Y}$ while the true label is Y. In classification the loss function is often a zero-one loss, that is, $\ell(Y, \hat{Y})$ is zero when the predicted label matches the true label $Y = \hat{Y}$ and one otherwise. To find our best classifier, we look for the classifier with the smallest expected loss. In other words, we look up the function f that minimizes the expected ℓ-risk, given by $\mathcal{R}_\ell(h) = \mathbf{E}_{X \times Y}[\ell(Y, f(X))]$.

Another naive approach is to minimize the empirical classification error $\mathbb{E}[\mathbb{1}_{\{-Yf(X) \geq 0\}}]$. To bypass the non convexity of $\mathbb{1}_{\mathbb{R}_+}$, we use convex risk minimization (CRM) Boucheron et al. [2005]. CRM defines a convex surrogate for the classification problem, called the cost function $\varphi : \mathbb{R} \mapsto \mathbb{R}_+$ convex, non-decreasing such that $\varphi(0) = 1$, hence $\varphi \geq \mathbb{1}_{\mathbb{R}_+}$. The classification problem consists in minimizing the expected φ-risk : $\mathbb{E}[\varphi(-Yf(X))]$. Typical loss functions are

- square loss: $\varphi(u) = (1 + u)^2$ for $u \geq 0$ and $\varphi(u) = 1$ for $u \leq 0$, leading to regression methods.
- perplexity loss: $\varphi(u) = \log(e(1 + u)))$ for $u > -1$ and $\varphi(u) = -\infty$ for $u \leq -1$.
- logit loss: $\varphi(u) = \log_2(1 + e^u)$, leading to logistic regression methods and cross entropy.
- hinge loss: $\varphi(u) = max(0, 1 + u)$. This leads to SVM methods.
- exponential loss: $\varphi(u) = e^u$.

We have also a loss function defined directly between Y and $\hat{Y}$ as follows:

- mean square error: $\ell(Y, f(X)) = (Y - f(X))^2$ that leads to standard regression methods.
- mean absolute error: $\ell(Y, f(X)) = |Y - f(X)|$.

- cross entropy: $\ell(Y, f(X)) = -\frac{1+Y}{2} \log \frac{1+f(X)}{2}$ leading to logistic regression with the usual convention $0 \log 0 = 0$ justified by $\lim_{x \to 0^+} x \log x = 0$ (trivially obtained by L'Hopital's rule).

- Huber loss: $\ell(Y, f(X)) = \frac{1}{2}(Y - f(X))^2$ if $|Y - f(X)| \leq \delta$ and $\ell(Y, f(X)) = \delta(|Y - f(X)| - \frac{1}{2}\delta^2$

We see from above that for SL there are numerous loss functions and that cross entropy is one criterion among others. We will see that this cross entropy has a nice interpretation in RL.

5.4.2 Reinforcement Learning Background

RL is usually modeled by an agent that interacts with an environment $\mathcal{E}$ over a number of discrete time steps. At each time step t, the agent levies a state s_t and picks an action a_t from a set of possible actions $\mathcal{A}$. This choice is done according to its policy π, where π is a mapping from states s_t to actions a_t. Once the action is decided and executed, the agent levies the next state s_{t+1} and a scalar reward r_t. This goes on until the agent reaches a terminal state. The expected cumulated discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the sum of accumulated returns, where at each time step, future returns are discounted with the discount factor $\gamma \in (0, 1]$. At time t, a rational agent seeks to maximize its expected return given his current state s_t.

We traditionally define

- the value of state s under policy π is defined as $V^\pi(s) = \mathbb{E}[R_t | s_t = s]$ and is simply the expected return for following policy π from state s ([Watkins, 1989]).

- the action value function under policy π $Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a]$ is defined as the expected return for selecting action a in state s and following policy π (Williams [1992]).

Both the optimal value function $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ and the optimal value of state $V^*(s) = \max_\pi V^\pi(s)$ satisfy Bellmann equations.

Whenever states and actions are too large, we are forced to represent the action value function with a function approximator, such as a neural network. Denoting the parameters θ, the state action function writes $Q(s, a; \theta)$

The updates to θ can be derived from a variety of reinforcement learning algorithms. In particular, in value-based methods, policy-based model-free methods directly parameterize the policy $\pi(a|s; \theta)$ and update the parameters θ by performing, typically approximate, gradient ascent on $\mathbb{E}[R_t]$.

An illustration of such a method is REINFORCE due to Williams [1992]. Standard REINFORCE updates the policy parameters θ in the direction $R_t \nabla_\theta \log \pi(a_t|s_t; \theta)$, which is an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$. It is possible to reduce the variance of this estimate while keeping it unbiased by subtracting a learned function of the state $b_t(s_t)$, known as a baseline [Williams, 1992], from the return. The resulting gradient is $\nabla_\theta \log \pi(a_t|s_t; \theta)(R_t - b_t(s_t))$. This approach can be viewed as an actor-critic architecture where the policy π is the actor and the baseline b_t is the critic[Sutton and Barto, 1998, Degris et al., 2012].

We now prove a new formulation of REINFORCE.

Proposition 5.1. *The gradient descent in REINFORCE can also be computed by minimizing the*

following quantity

$$\widetilde{J}(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} R_i(\tau) \log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{5.1}$$

For Advantage Actor Critic method, the gradient descent can also be computed by minimizing the following quantity:

$$\widetilde{J}(\theta) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} A(s_{i,t}, a_{i,t}) \log \pi_\theta(a_{i,t} \mid s_{i,t})}{N} \tag{5.2}$$

Proof. We provide here a quick proof of REINFORCE with modern notations. Let us denote by $r(s_t, a_t)$ the reward for a state s_t and action a_t. Let us assume we have some time horizon (that may be either finite or infinite). Let us denote by $\tau = (s_1, a_1, \ldots, s_T, a_T)$ a trajectory generated by our policy approximator governed by a parameter θ. Using the Markov property of our MDP process, the probability of a given trajectory $\mathbb{P}(\tau|\theta)$ can be decomposed into a product of conditional probabilities as follows:

$$\mathbb{P}(\tau|\theta) = \mathbb{P}(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) \mathbb{P}(s_{t+1} \mid s_t, a_t) \tag{5.3}$$

Taking the log of the above equation (5.3), using the fact that the log of a product is the sum of the logs, we get:

$$\log \mathbb{P}(\tau|\theta) \quad = \quad \log \mathbb{P}(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid s_t) + \log \mathbb{P}(s_{t+1} \mid s_t, a_t) \tag{5.4}$$

Differentiating with respect to theta gives (since most of the terms do not depend on θ):

$$\nabla_\theta \log \mathbb{P}(\tau|\theta) \quad = \quad \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \tag{5.5}$$

Recall we want to minimize the expected return, $J(\theta)$, defined as

$$J(\theta) = \mathbb{E}_{\tau\sim\mathbb{P}(\tau|\theta)} \left[\sum_{t=1}^{T} r(s_t, a_t) \right] = \int_\tau R(\tau) \mathbb{P}(\tau|\theta) d\tau \tag{5.6}$$

The above notation $\tau \sim \mathbb{P}(\tau|\theta)$ indicates that we're sampling trajectories τ from the probability distribution of our policy approximator governed by θ and $R(\tau)$ is the sum of all the future (discounted) rewards.

To find the optimal θ, we do gradient descent and hence need to compute

$$\nabla_\theta J(\theta) \quad = \quad \nabla_\theta \int_\tau R(\tau) \mathbb{P}(\tau|\theta) d\tau$$

Using the fact that we can interchange integral and expectation (5.7), assuming smooth functions and Lebesgue dominated convergence to justify that we can bring the gradient under the integral, we get:

$$\nabla_\theta J(\theta) = \int_\tau R(\tau)\nabla_\theta \mathbb{P}(\tau|\theta)d\tau \tag{5.7}$$

As the log gradient of a function is the quotient of the gradient and the function, we have:

$$\nabla_\theta J(\theta) = \int_\tau R(\tau)\nabla_\theta \log \mathbb{P}(\tau|\theta)\mathbb{P}(\tau|\theta)d\tau \tag{5.8}$$

Expressed the integral as an expectation, we conclude:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[R(\tau)\nabla_\theta \log \mathbb{P}(\tau|\theta)\right] \tag{5.9}$$

where in the above equations, we have first used that we can interchange integral and expectation (5.7), assuming smooth function and Lebesgue dominated convergence to justify that we can bring the gradient under the integral, then in (5.8), we have used the fact that the log gradient of a function is the quotient of the gradient and the function, and finally have expressed the integral as an expectation. Finally, using the fact that the gradient of the log probability of the trajectory is the sum of the gradient of the log policy probabilities (5.5), we obtain the final expression:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[R(\tau)\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \tag{5.10}$$

To turn this into something tractable, just express this as a Monte Carlo sum as follows:

$$\nabla_\theta J(\theta) = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} R_i(\tau)\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{5.11}$$

As the RHS does only depend on θ in the term $\log \pi_\theta(a_{i,t} \mid s_{i,t})$, minimizing $J(\theta)$ is the same as minimizing $\widetilde{J}(\theta)$ given by:

$$\widetilde{J}(\theta) = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} R_i(\tau)\log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{5.12}$$

As for Advantage Actor Critic, the above proof is the same and leads to the result. □ □

It is enlightening to see that the two formulations, traditional reinforce and actor critic are very close to SL method with cross entropy. Recall that cross entropy is given by $Y\log(\hat{Y})$ for labels with value in $\{0, 1\}$. This leads to the table 5.1

Last but not least, recall that there is a connection between cross entropy and Kullback Leibler divergence. Recall that the cross entropy for the distributions p and q over a given set is defined as follows:

$$H(p, q) = \mathbb{E}_p[-\log q] \tag{5.13}$$

Term	SL	RL (REINFORCE)	RL (A2C)
true label	Y	expected future rewards: $R(\tau)$	expected advantage: $A(s,a) = Q(s,a) - V(s)$
log term	$\log(\hat{Y})$	log of policy: $\log \pi_\theta(a_{i,t} \mid s_{i,t})$	log of policy: $\log \pi_\theta(a_{i,t} \mid s_{i,t})$
cross entropy	$\frac{\sum_{i=1}^{N} Y_i \log \hat{Y}_i}{N}$	Monte Carlo expectation: $\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} R_i(\tau) \log \pi_\theta(a_{i,t}\|s_{i,t})}{N}$	Monte Carlo expectation: $\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} A(s_{i,t},a_{i,t}) \log \pi_\theta(a_{i,t}\|s_{i,t})}{N}$

Table 5.1: Comparing SL and RL for REINFORCE and A2C methods

There is a straightforward connection to the Kullback–Leibler divergence $D_{\mathrm{KL}}(p\|q)$ of q from p, sometimes referred to as the relative entropy of p with respect to q given by

$$H(p,q) = H(p) + D_{\mathrm{KL}}(p\|q) \tag{5.14}$$

where $H(p)$ is the entropy of p. As the entropy term $H(p)$ is constant given the true distribution p, minimizing the cross entropy or the Kullback Leibler divergence is equivalent. However, for RL, this gives another nice interpretation. As shown previously, PGM can be cast as a cross entropy minimization program. Since the difference between Kullback Leibler divergence and cross entropy is this entropy term, it makes sense to incorporate this entropy term in our gradient descent optimization. In theory, the entropy term should be multiplied by one. However in practice as the entropy term is estimated by the empirical entropy, it makes sense somehow to multiply the entropy term by a regularization term λ, leading to a variation of PGMs where instead of computing a gradient on cross entropy as in REINFORCE, we add an additional term called an entropy regularization. Somehow, this changes the minimization problem to something that is a modified Kullback Leibler divergence. The cross entropy term itself is computed on the future expected reward. The analogy holds between SL and RL as long as actions do not influence or slightly influence the environment.

Remark 5.2. As a matter of fact, our tight connection between RL and SL makes a lot of sense and is very intuitive. Indeed, SL can be cast as an optimization problem, which can be solved by gradient descent (that's with stochastic gradient descent (SGD) and backprop in neural networks). Value estimation and value optimization in RL can also be cast as an optimization problem. It can also be solved by gradient descent (with TD-learning or Q-learning). Finally, policy optimization can also be solved by gradient descent (that's policy gradient). The only difference is really the loss function that is optimized.

5.5 Experiments

We will apply our remark to a very specific environment where actions do not influence the environment. The considered reinforcement problem is a financial trading game concerning

the Facebook stock (data were retrieved from https://finance.yahoo.com/quote/FB. We denote by $(P_t)_{t=1,...}$ the daily closing price of the Facebook stock in sequential order. For each day, we compute the daily return as follows $r_t = \frac{P_t}{P_{t-1}} - 1$

The environment is composed of the n last daily returns. We intentionally assume n last returns to emphasize that the choice of taking $n = 5$ (last week) or $n = 10$ (last two weeks) or $n = 20$ (last month) is a model design decision. As returns are continuous, our state space is $\mathbb{R}^n_+$, which by RL standard is very large. Our possible actions are each day threefold: either do nothing, buy or sell the Facebook stock. If we decide to enter a new position at time t, this will only be materialized the next day and hence we will be initially having an open position only at time $t + 1$ initialized at the entering price p_{t+1}. Hence, if we only keep the position for one period, we will be facing the return r_{t+2} as our position will be only closed at time $t + 2$.

As for the reward, we take the Sharpe ratio of the trading strategy. As we compute the Sharpe ratio with daily returns, to compute the annual Sharpe ratio, we multiply the daily Sharpe ratio by a scaling factor equal to $\sqrt{250}$, assuming 250 trading days per year. We compute the daily Sharpe ratio as the mean of daily returns over their standard deviations. This implies in particular that we implicitly take a benchmark rate in the Sharpe ratio equal to 0.

To compute our Mark to Market (the value of our trading strategy), we mark any open position to the last known price. We parametrize our policy with a deep network consisting of two fully connected layers composed of 16 ReLU nodes. We use ReLU activation function (as opposed to sigmoid) for two main reasons: sparsity and less chance to incur vanishing gradient. Recall a ReLU is defined as $h = max(0, a)$ where $a = Wx + b$. Vanishing gradient has less chance to vanish since the gradient is either zero or constant whenever the linear term is strictly positive ($a > 0$). In contrast, the gradient of sigmoid becomes increasingly small when the absolute value of x tends to be large. This constant character of the ReLU's gradient results in faster learning. The other benefit of ReLUs is sparsity. Sparsity arises when the linear term is non-positive: $a \leq 0$. The more units with non-positive terms exist in a layer, the more sparse the resulting representation. In contrast, sigmoids always generate some non-zero value resulting in dense representations.

In our experience, we take two fully connected layers composed of 16 ReLU nodes and use A2C method to solve this reinforcement learning problem. For exploration purposes, we use three epsilon greedy method with epsilon equal to 5%, 25% and 50%. We compare these constant learning rates to two annealing method that decreases linearly the learning rate from 1 to 0.1% and from 0.5 to 0.1%. We provide the overall achieved Sharpe ratio with the Sharpe ratio reward choice for various learning rates and annealing rates. We obtained an overall Sharpe ratio higher than 1 which is quite a nice achievement compared to traditional investment strategies that typically do not perform better than a Sharpe ratio of 1. We see in the resulting experience that the exploration methods that work best are the two epsilon greedy annealed methods.

We can notice in figure 5.1 that the overall Sharpe ratio is above 3 which is very high by financial market standards. This could be explained by the fact that the Facebook stock has been incredibly raising over the last five years. Hence the algorithm has not much difficulty finding the optimal strategy that is to buy and hold the stock

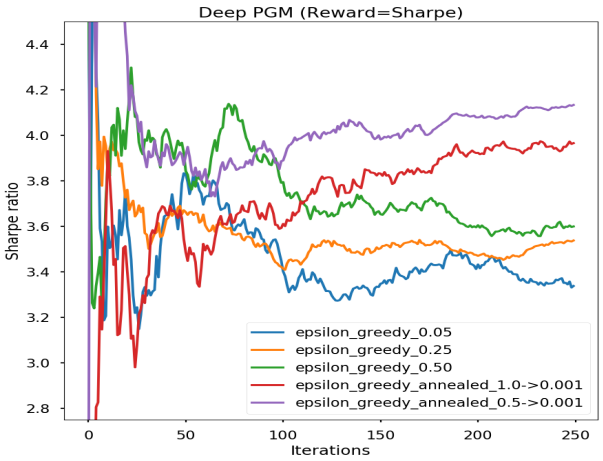


Figure 5.1: Comparison of various learning rate strategies for our experience. The best strategies are annealing learning rates. These strategies work well as they have the right balance between exploration-exploitation. At first, as their learning rate is quite high, they explore more than their fixed learning rate counterpart. As we progress in the algorithm, the learning rate decreases and this strategy progressively shifts from exploration to exploitation.

5.6 Summary

We show in this chapter that there are tight connections between SL and RL. PGM in RL can be cast as cross entropy minimization problems where true labels are replaced by expected future reward or advantage while the log term is changed into the log policy term. This analogy takes its root from the minimization problem where we are looking for the parameters that maximize the expected future reward or advantage. Should this optimization objective changed, we conjecture that we could make other analogies between SL and RL.

This chapter concludes the model-free part where we have aimed to use machine learning alone to solve the portfolio allocation problem. Like other chapters of the thesis, this chapter has been the subject of a publication in a conference Benhamou and Saltiel [2020].

PART III

Understanding the model

Overview

In this last part, we present two works that help understand better the machinery behind Machine Learning.

One of the limitations of the adoption of Machine Learning in quantitative finance is its Black Box aspect. Compared to traditional quantitative methods, ML can use a large set of parameters and hyperparameters. Because of the larger number of parameters and hyperparameters, ML without care is more prone to overfitting than traditional quantitative methods. In addition, hyper parameters determination relies on non convex optimization methods that can also seem mysterious.

In this last part, we first examine the covariance matrix adaptation evolution strategy (CMA-ES) which is one of the state of the art methods for fine-tuning hyper parameters in Machine Learning. In chapter 6, we prove that this optimization approach called black box optimization or gradient-free optimization can be reformulated as a Bayesian optimization. This is interesting as it naturally leads to a new algorithm formulation called Bayesian CMAES very similar to the initial CMAES method. It also shows that Bayesian optimization and Black box optimization are very similar.

We conclude this part by presenting some global explanation methods for variables in machine learning called Shapley values. In chapter 7, we illustrate how Shapley values can be used to get more intuition about the rules learned by a model in a LightGBM model. Furthermore, we can rank the Shapley values by order of magnitude importance, defined as the average absolute Shapley values over the training set of the model to rank features and automatically detect the most important variables. The average Shapley values are essentially the average impact on model output when a feature becomes "hidden" from the model. Furthermore, examining the correlation of a feature to its Shapley value provides insight into the effect of this feature on the probability of a stock market crash.

6 Bayesian CMAES

6.1 Introduction

The covariance matrix adaptation evolution strategy (CMA-ES) Hansen and Ostermeier [2001] is arguably one of the most powerful real-valued derivative-free optimization algorithms, finding many applications in machine learning. It is a state-of-the-art optimizer for continuous black-box functions as shown by the various benchmarks of the COmparing Continuous Optimisers (http://coco.gforge.inria.fr) INRIA platform for ill-posed functions. It has led to a large number of papers and articles and we refer the interested reader to Hansen and Ostermeier [2001], Auger et al. [2004], Igel et al. [2007], Auger and Hansen [2009], Hansen and Auger [2011], Auger and Hansen [2012], Hansen and Auger [2014], Akimoto et al. [2015, 2016], Ollivier et al. [2017] and Varelas et al. [2018] to cite a few.

It has been successfully applied in many unbiased performance comparisons and numerous real-world applications. In particular, in machine learning, it has been used for direct policy search in reinforcement learning and hyper-parameter tuning in supervised learning Igel et al. [2009], Heidrich-Meisner and Igel [2009], Igel [2010], and references therein, as well as hyperparameter optimization of deep neural networks Loshchilov and Hutter [2016]

In a nutshell, the (μ / λ) CMA-ES is an iterative black box optimization algorithm, that, in each of its iterations, samples λ candidate solutions from a multivariate normal distribution, evaluates these solutions (sequentially or in parallel) retains μ candidates and adjusts the sampling distribution used for the next iteration to give a higher probability to good samples. Each iteration can be individually seen as taking an initial guess or *prior* for the multivariate parameters, namely the mean and the covariance, and after making an experiment by evaluating these sample points with the fit function updating the initial parameters accordingly.

Historically, the CMA-ES has been developed heuristically, mainly by conducting experimental research and validating intuitions empirically. The research was done without much focus on theoretical foundations because of the apparent complexity of this algorithm. It was only recently that Akimoto et al. [2010], Glasmachers et al. [2010] and Ollivier et al. [2017] made a breakthrough and provided a theoretical justification of CMA-ES updates thanks to information geometry. They proved that CMA-ES was performing natural gradient descent in the Fisher information metric.

These works provided a nice explanation for the reasons for the performance of the CMA-ES because of strong invariance properties and good search directions.

There is however another way of explanation that has been so far ignored and could also bring nice insights about CMA-ES. It is Bayesian statistics theory. At light of Bayesian statistics, CMA-ES can be seen as an iterative prior posterior update. But there is some real complexity due to tricky updates that may explain why this has always been passed by. First of all, in a regular Bayesian approach, all sample points should be taken. This is not the case in the (μ/λ) CMA-ES, as out of λ generated paths, only the μ best are kept. Updating weights are also constant which is not consistent with Bayesian updates. But more importantly, the covariance matrix update does not read easily as a Bayesian prior posterior update. The update is split between a weighted combination of a rank one matrix referred to $p_C p_C^T$ with parameter c_1 and a rank $min(\mu, n)$ matrix with parameter c_μ, whose details are given for instance in Hansen [2016]. These two updates of the covariance matrix make the Bayesian rendering challenging as these updates are done according to two paths: the isotropic and anisotropic evolution ones. All this may explain why interpreting the CMA-ES algorithm as a Bayesian algorithm seemed a daunting task and was not tackled before.

This is precisely the objective of this chapter. Section 6.2 recalls various Bayesian concepts for prior and posterior to highlight the analogy of an iterative Bayesian update. Section 6.3 presents in greater detail the Bayesian approach of CMA-ES, with the corresponding family of derived algorithms, emphasizing the various design choices that can lead to multiple algorithms. Section 6.4 provides numerical experiments and shows that Bayesian adapted CMA-ES algorithms perform well on convex and non convex functions and are comparable to the classical CMA-ES. We finally conclude with some possible extensions and further experiments.

6.2 Framework

CMA-ES computes at each step an update for the mean and covariance of the distribution of the minimum. From a very general point of view, this can be interpreted as a prior posterior update in Bayesian statistics.

6.2.1 Bayesian vs Frequentist probability theory

Let us recall some basic results from Bayesian statistics. The justification of the Bayesian approach is discussed in Robert [2007]. In Bayesian probability theory, we assume a distribution on unknown parameters of a statistical model. This leads to an axiomatic reduction from the notion of the unknown to the notion of randomness but with probability. We do not know the value of the parameters for sure but we know specific values that these parameters can take with high probabilities. This creates a prior distribution that is updated as we make some experiments as shown in Gelman et al. [2004], Marin and Robert [2007], Robert [2007]. Following Jordan [2010], we can mention the DeFinetti theorem that provides a real justification for the existence of a prior. This relies on infinite exchangeability, which is a weaker concept of i.i.d.

Definition 6.1. *(Infinite exchangeability). We say that $(x_1, x_2, ...)$ is an infinitely exchangeable sequence of random variables if, for any n, the joint probability $p(x_1, x_2, ..., x_n)$ is invariant to permutation of the indices. That is, for any permutation π,*

$$p(x_1, x_2, ..., x_n) = p(x_{\pi 1}, x_{\pi 2}, ..., x_{\pi n})$$

Equipped with this definition, the De Finetti's theorem as provided below states that exchangeable observations are conditionally independent relative to some latent variable.

Theorem 6.2. *(De Finetti, 1930s). A sequence of random variables* $(x_1, x_2, ...)$ *is infinitely exchangeable iff, for all n,*

$$p(x_1, x_2, ..., x_n) = \int \prod_{i=1}^{n} p(x_i|\theta)P(d\theta),$$

for some measure P on θ*.*

This representation theorem 6.2 justifies the use of priors on parameters since for exchangeable data, there must exist a parameter θ, a likelihood $p(x|\theta)$ and a distribution π on θ. A proof of De Finetti theorem is for instance given in Schervish [1996] (section 1.5).

6.2.2 Conjugate priors

In Bayesian statistical inference, the probability distribution that expresses one's (subjective) beliefs about the distribution parameters before any evidence is taken into account, is called *the prior* probability distribution, often simply called the prior. In CMA-ES, it is the distribution of the mean and covariance. We can then update our prior distribution with the data using Bayes' theorem to obtain a posterior distribution. The *posterior* distribution is a probability distribution that represents the updated beliefs about the parameters after having seen the data. The Bayes' theorem tells us *the fundamental rule* of Bayesian statistics, that is

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}$$

The proportional sign ($\propto$) indicates that one should compute the distribution up to a renormalization constant that enforces the distribution sums to one. This rule is simply a direct consequence of Bayes' theorem. Mathematically, let us say that for a random variable X, its distribution p depends on a parameter θ that can be multi-dimensional. To emphasize the dependency of the distribution on the parameters, let us write this distribution as $p(x|\theta)$ and let us assume we have access to a prior distribution $\pi(\theta)$. Then the joint distribution of (θ, x) writes simply as

$$\phi(\theta, x) = p(x|\theta)\pi(\theta)$$

The marginal distribution of x is trivially given by marginalizing (or summing) the joint distribution over θ as follows:

$$m(x) = \int \phi(\theta, x)d\theta = \int p(x|\theta)\pi(\theta)d\theta$$

The posterior of θ is obtained by Bayes' formula as

$$\pi(\theta|x) = \frac{p(x|\theta)\pi(\theta)}{\int p(x|\theta)\pi(\theta)d\theta} \propto p(x|\theta)\pi(\theta) \tag{6.1}$$

Computing a posterior is tricky and does not bring much value in general. A key concept in Bayesian statistics is conjugate priors that make the computation really easy and is described at length below.

Definition 6.3. *A prior distribution $\pi(\theta)$ is said to be a conjugate prior if the posterior distribution $\pi(\theta|x)$ remains in the same distribution family as the prior.*

At this stage, it is relevant to introduce exponential family distributions as this higher level of abstraction that encompasses the multivariate normal trivially solves the issue of founding conjugate priors. This will be very helpful for inferring conjugate priors for the multivariate Gaussian used in CMA-ES.

Definition 6.4. *A distribution is said to belong to the exponential family if it can be written (in its canonical form) as:*

$$p(\mathbf{x}|\eta) = h(\mathbf{x}) \exp\{\eta \cdot T(\mathbf{x}) - A(\eta)\}, \tag{6.2}$$

where η is the natural parameter, $T(\mathbf{x})$ is the sufficient statistic, $A(\eta)$ is the log-partition function and $h(\mathbf{x})$ is the base measure. η and $T(\mathbf{x})$ may be vector-valued. Here $a \cdot b$ denotes the inner product of a and b.

Using the fact that the probability sums to one, the log-partition function can be recovered by the integral

$$A(\eta) = \log \int_{\mathcal{X}} h(\mathbf{x}) \exp(\eta \cdot T(\mathbf{x}))\, \mathrm{d}x. \tag{6.3}$$

One defines the natural parameter space as $\Omega = \{\eta \in \mathbb{R}^m | A(\theta) < +\infty\}$. Obviously, $\eta \in \Omega$. In addition, it can be shown that Ω is a convex set and that $A(\cdot)$ is a convex function on Ω.

Remark 6.5. Not surprisingly, the normal distribution $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and covariance matrix Σ belongs to the exponential family but with a different parametrization. Its exponential family form is given by:

$$\eta(\mu, \Sigma) = \begin{bmatrix} \Sigma^{-1}\mu \\ \mathrm{vec}(\Sigma^{-1}) \end{bmatrix}, \qquad T(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ \mathrm{vec}(-\frac{1}{2}\mathbf{x}\mathbf{x}^{\mathrm{T}}) \end{bmatrix}, \tag{6.4a}$$

$$h(\mathbf{x}) = (2\pi)^{-\frac{d}{2}}, \qquad A(\eta(\mu, \Sigma)) = \frac{1}{2}\mu^{\mathrm{T}}\Sigma^{-1}\mu + \frac{1}{2}\log|\Sigma|. \tag{6.4b}$$

where in equations (6.4a), the notation vec$(\cdot)$ means we have vectorized the matrix, stacking each column on top of each other. Canonical parameters are different from traditional (also called moment) parameters.

For an exponential family distribution, it is particularly easy to form conjugate prior as explained by the proposition below:

Proposition 6.6. *If the observations have a density of the exponential family form $p(x|\theta, \lambda) = h(x) \exp\left(\eta(\theta, \lambda)^T T(x) - nA(\eta(\theta, \lambda))\right)$, with λ a set of hyper-parameters, then the prior with likelihood defined by $\pi(\theta) \propto \exp\left(\mu_1 \cdot \eta(\theta, \lambda) - \mu_0 A(\eta(\theta, \lambda))\right)$ with $\mu \triangleq (\mu_0, \mu_1)$ is a conjugate prior.*

The proof is given in the appendix subsection 6.5.1. As we can vary the parameterization of the likelihood, we can obtain multiple conjugate priors. Because of conjugacy, if the initial parameters of the multivariate Gaussian follow the prior, the posterior stays in the same family making the update of the parameters really easy. Said differently, with conjugate prior, we can easily compute the posterior accurately. As we get more information about the likelihood, our posterior distribution becomes peaked towards the true distribution as shown in figure6.1.

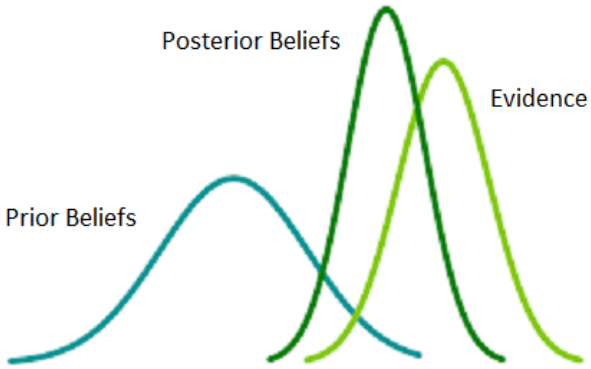


Figure 6.1: As we get more and more information using the likelihood, the posterior becomes more peak.

6.2.3 Optimal updates for NIW

The two natural conjugate priors for the Multivariate normal that update both the mean and the covariance are the normal-inverse-Wishart (if we want to update the mean and covariance parameters) or the normal-Wishart (if we are interested in updating the mean and the precision matrix, which is the inverse of the covariance matrix). In this chapter, we will stick to the normal-inverse-Wishart to keep things simple. The Normal-inverse-Wishart distribution is parametrized by $\boldsymbol{\mu}_0, \lambda, \boldsymbol{\Psi}, \nu$ and its distribution is given by

$$f(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \boldsymbol{\mu}_0, \lambda, \boldsymbol{\Psi}, \nu) = \mathcal{N}\left(\boldsymbol{\mu} \middle| \boldsymbol{\mu}_0, \frac{1}{\lambda}\boldsymbol{\Sigma}\right) \mathcal{W}^{-1}(\boldsymbol{\Sigma} | \boldsymbol{\Psi}, \nu)$$

where $\mathcal{W}^{-1}$ denotes the inverse Wishart distribution. The key theoretical guarantee of the BCMA-ES is to update the mean and covariance of our CMA-ES optimally as follows.

Proposition 6.7. *If our sampling density follows a d dimensional multivariate normal distribution* $\sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *with unknown mean* μ *and covariance* Σ *and if its parameters are distributed according to a Normal-Inverse-Wishart* $(\mu, \Sigma) \sim \text{NIW}(\mu, \kappa, \nu, \psi)$ *and if we observe* $\mathcal{X} = (x_1, .., x_n)$ *samples, then the posterior is also a Normal-Inverse-Wishart with different parameters* $\text{NIW}(\mu^\star, \kappa^\star, \nu^\star, \psi^\star)$ *given by*

$$\begin{aligned} \mu^\star &= \frac{\kappa\mu + n\overline{x}}{\kappa + n}, \\ \kappa^\star &= \kappa + n, \\ \nu^\star &= \nu + n \\ \psi^\star &= \psi + \sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T + \frac{\kappa n}{\kappa + n}(\overline{x} - \mu)(\overline{x} - \mu)^T \end{aligned} \tag{6.5}$$

with $\overline{x}$ *the sample mean.*

Remark 6.8. This proposition is the cornerstone of the BCMA-ES. It provides the theoretical guarantee that the updates of the parameters in the algorithm are accurate and optimal under the

assumption of the prior. In particular, this implies that any other formula for the update of the mean and variance and in particular the ones used in the mainstream CMA-ES assumes a different prior.

Proof. A proof is given in the appendix part 6.5.2 of this chapter. □

6.3 Bayesian CMA-ES

6.3.1 Main assumptions

Our main assumptions are the followings :

- The parameters of the multi-variate Gaussian follow a conjugate prior distribution, namely a Normal Inverse Wishart.
- The minimum of our objective function f follows a multi-variate normal law.

6.3.2 Simulating the minimum

One of the main challenges is to simulate the likelihood of inferring the posterior. The key question is really to use the additional information of the function value f for candidate points. At step t in our algorithm, we suppose multivariate Gaussian parameters μ and Σ follow a Normal Inverse Wishart denoted by $NIW(\mu_t, \kappa_t, \nu_t, \psi_t)$.

In full generality, we need to do a Monte Carlo of Monte Carlo as the parameters of our multivariate normal are themselves stochastic. However, we can simplify the problem and take their mean values. It is very effective in terms of computation and reduces Monte Carlo noise. For the Normal Inverse Wishart distribution, there exist closed-form expressions for these mean values given by:

$$\mathbb{E}_t[\mu] = \mu_t \tag{6.6}$$

and

$$\mathbb{E}_t[\Sigma] = \frac{\psi_t}{\nu_t - n - 1} \tag{6.7}$$

We simulate potential candidates $\mathcal{X} = \{X_i\} \sim \mathcal{N}(\mathbb{E}_t[\mu], \mathbb{E}_t[\Sigma])$ and evaluate them $f(X_i)$. If the distribution of the minimum was correct, the minimum would concentrate around $\mathbb{E}_t[\mu]$ and be spread with a variance of $\mathbb{E}_t[\Sigma]$. When evaluating potential candidates, as our guess is not right, we do not get values centered around $\mathbb{E}_t[\mu]$ and spread with a variance of $\mathbb{E}_t[\Sigma]$. This comes from three things:

- Our assumed minimum is not right. We need to shift our normal to the right minimum!
- Our assumed variance is not right. We need to compute it on real data taken into additional information given by f.
- Last but not least, our Monte Carlo simulation adds some random noise.

For the last issue, we can correct any of our estimators by the Monte Carlo bias. This can be done using standard control variate as the simulated mean and variance are given: $\mathbb{E}_t[\mu]$ and $\mathbb{E}_t[\Sigma]$ respectively and we can compute for each of them the bias explicitly.

The first two issues are more complex. Let us tackle each issue one by one.

To recover the true minimum, we design two strategies.

- We design a strategy where we rebuild our normal distribution but use sorted information of our X's weighted by their normal density to ensure this is a true normal corrected from the Monte Carlo bias. We need to explicitly compute the weights. For each simulated point X_i, we compute it assumed density denoted by $d_i = \mathcal{N}(\mathbb{E}_t[\mu], \mathbb{E}_t[\Sigma])(X_i)$ where $\mathcal{N}(\mathbb{E}_t[\mu], \mathbb{E}_t[\Sigma])(.)$ denotes the p.d.f. of the multivariate Gaussian.

  We divide these densities by their sum to get weights $(w_i)_{i=1..k}$ that are positive and sum to one as follows. $w_j = d_j / \sum_{i=1}^{k} d_i$. Hence for k simulated points, we get $\{X_i, w_i\}_{i=1..k}$. We reorder jointly the uplets (points and density) in terms of their weights in decreasing order.

  To insist we take sorted value in decreasing order with respect to the weights $(w_i)_{i=1..k}$, we denote the order statistics $(i), w \downarrow$.

  This first sorting leads to k new uplets $\{X_{(i),w\downarrow}, w_{(i),w\downarrow}\}_{i=1..k}$. Using a *stable* sort (that keeps the order of the density), we sort jointly the uplets (points and weights) according to their objective function value (in increasing order this time) and get k new uplets $\{X_{(i),f\uparrow}, w_{(i),w\downarrow}\}_{i=1..k}$. We can now compute the empirical mean $\overline{\mu}_t$ as follows:

$$\overline{\mu}_t = \underbrace{\sum_{i=1}^{k} w_{(i),w\downarrow} \cdot X_{(i),f\uparrow}}_{\text{MC mean for } X_{f\uparrow}} - \underbrace{\left(\sum_{i=1}^{k} w_i X_i - \overline{\mu}_t\right)}_{\text{MC bias for } X} \tag{6.8}$$

  The intuition of equation (6.8) is to compute in the left term the Monte Carlo mean using reordered points according to their objective value and correct our initial computation by the Monte Carlo bias computed as the right term, equal to the initial Monte Carlo mean minus the real mean. We call this strategy one.

- If we think for a minute about the strategy one, we get the intuition that when starting the minimization, it may not be optimal. This is because weights are proportional to $\exp\left\{\frac{1}{2}(X - \mathbb{E}_t[\mu])^T(\mathbb{E}_t[\Sigma])^{-1}(X - \mathbb{E}_t[\mu])\right\}$.

  When we start the algorithm, we use a large search space, hence a large covariance matrix $\overline{\Sigma}_t$ which leads to weights that are quite similar. Hence even if we sort candidates by their fit, ranking them according to the value of f in increasing order, we will move our theoretical multivariate Gaussian little by little. A better solution is to brutally move the center of our multivariate Gaussian to the best candidate seen so far, as follows:

$$\overline{\mu}_t = \arg\min_{X \in \mathcal{X}} f(X) \tag{6.9}$$

  We call this strategy two. Intuitively, strategy two should work better when starting the algorithm while strategy one should work better once we are close to the solution.

To recover the true variance, we can adapt what we did in strategy one as follows:

- $$\overline{\Sigma}_t = \underbrace{\sum_{i=1}^{k} w_{(i),w\downarrow} \cdot \left(X_{(i),f\uparrow} - \overline{X}_{(.),f\uparrow}\right)\left(X_{(i),f\uparrow} - \overline{X}_{(.),f\uparrow}\right)^T}_{\text{MC covariance for } X_{f\uparrow}} - \underbrace{\left(\sum_{i=1}^{k} w_i \cdot \left(X_i - \overline{X}\right)\left(X_i - \overline{X}\right)^T - \overline{\Sigma}_t\right)}_{\text{MC covariance for simulated } X} \tag{6.10}$$

  where $\overline{X}_{(.),f\uparrow} = \sum_{i=1}^{k} w_{(i),w\downarrow} X_{(i),f\uparrow}$ and $\overline{X} = \sum_{i=1}^{k} w_i X_i$ are respectively the mean of the sorted and non sorted points.

- Again, we could design another strategy that takes part of the points but we leave this to further research.

Once we have the likelihood mean and variance using (6.9) and (6.10) or (6.8) and (6.10), we update the posterior law according to equation (6.5). This gives us the iterative conjugate prior parameters updates:

$$\begin{aligned} \mu_{t+1} &= \frac{\kappa_t \mu_t + n\overline{\mu}_t}{\kappa_t + n}, \\ \kappa_{t+1} &= \kappa_t + n, \\ \nu_{t+1} &= \nu_t + n, \\ \psi_{t+1} &= \psi_t + \overline{\Sigma}_t + \frac{\kappa_t n}{\kappa_t + n}\left(\overline{\mu}_t - \mu_t\right)\left(\overline{\mu}_t - \mu_t\right)^T \end{aligned} \tag{6.11}$$

The resulting algorithm is summarized in Algo 4.

Proposition 6.9. *Under the assumption of a NIW prior, the updates of the BCMA-ES parameters for the expected mean and variance write as a weighted combination of the prior expected mean and variance and the empirical mean and variance as follows*

$$\begin{aligned} \mathbb{E}_{t+1}[\mu] &= \mathbb{E}_t[\mu] + w_t^{\mu}\left(\overline{\mu}_t - \mathbb{E}_t[\mu]\right), \\ \mathbb{E}_{t+1}[\Sigma] &= \underbrace{w_t^{\Sigma,1}}_{\textit{discount factor}} \mathbb{E}_t[\Sigma] + w_t^{\Sigma,2} \underbrace{\left(\overline{\mu}_t - \mathbb{E}_t[\mu]\right)\left(\overline{\mu}_t - \mathbb{E}_t[\mu]\right)^T}_{\textit{rank one matrix}} \\ &\quad + w_t^{\Sigma,3} \underbrace{\overline{\Sigma}_t}_{\textit{rank (n-1) matrix}} \end{aligned}$$

$$\text{where} \quad \begin{aligned} w_t^{\mu} &= \frac{n}{\kappa_t + n}, \\ w_t^{\Sigma,1} &= \frac{\nu_t - n - 1}{\nu_t - 1}, \\ w_t^{\Sigma,2} &= \frac{\kappa_t n}{(\kappa_t + n)(\nu_t - 1)}, \\ w_t^{\Sigma,3} &= \frac{1}{\nu_t - 1} \end{aligned} \tag{6.12}$$

Remark 6.10. The proposition above is quite fundamental. It justifies that under the assumption of NIW prior, the update is a weighted sum of the previous expected mean and covariance. It is striking that it provides very similar formulae to the standard CMA-ES update. Recall that these updates given for the mean m_t and covariance C_t can be written as follows:

$$\begin{aligned} m_{t+1} &= m_t + \sum_{i=1}^{\mu} w_i \, (x_{i:\lambda} - m_t) \\ C_{t+1} &= \underbrace{(1 - c_1 - c_\mu + c_s)}_{\text{discount factor}} \, C_t + c_1 \underbrace{p_c p_c^T}_{\text{rank one matrix}} \\ &\quad + \underbrace{c_\mu \sum_{i=1}^{\mu} w_i \frac{x_{i:\lambda} - m_k}{\sigma_k} \left(\frac{x_{i:\lambda} - m_t}{\sigma_t} \right)^T}_{\text{rank } \min(\mu, n-1) \text{ matrix}} \end{aligned} \tag{6.13}$$

where the notations $m_t, w_i, x_{i:\lambda}, C_t, c_1, c_\mu, c_s$, etc... are given for instance in Wikipedia [2018].

Proof. See 6.5.3 in the appendix part of the chapter. □

Algorithm 4 Predict and Correct parameters at step t

1: **Simulate candidate**
2: Use mean values $\mathbb{E}_t[\mu] = \mu_t$ and $\overline{\Sigma}_t = \mathbb{E}[\Sigma] = \psi_t/(\nu_t - n - 1)$
3: Simulate k points $\mathcal{X} = \{X_i\}_{i=1..k} \sim \mathcal{N}(\mathbb{E}_t[\mu], \overline{\Sigma}_t)$
4: Compute densities $(d_i)_{i=1..k} = (\mathcal{N}(\mathbb{E}_t[\mu], \overline{\Sigma}_t)(X_i))_{i=1..k}$
5: Sort in decreasing order with respect to d to get $\{X_{(i),d\downarrow}, d_{(i),d\downarrow}\}_{i=1..k}$
6: Stable Sort in increasing order order with respect to $f(X_i)$ to get $\{X_{(i),f\uparrow}, d_{(i),d\downarrow}\}_{i=1..k}$
7: **Correct $\mathbb{E}_t[\mu]$ and $\overline{\Sigma}_t$**
8: Either Update $\mathbb{E}_t[\mu]$ and $\overline{\Sigma}_t$ using (6.8) and (6.10) **(strategy one)**
9: Or Update $\mathbb{E}_t[\mu]$ and $\overline{\Sigma}_t$ using (6.9) and (6.10) **(strategy two)**
10: Update $\mu_{t+1}, \kappa_{t+1}, \nu_{t+1}, \psi_{t+1}$ using (6.11)

6.3.3 Particularities of Bayesian CMA-ES

There are some subtleties that need to be emphasized.

- Although we assume a prior, we do not need to simulate the prior but can at each step use the expected value of the prior which means that we do not consume additional simulations compared to the standard CMA-ES.
- We need to tackle the local minimum (we will give an example of this in the numerical section) to avoid being trapped in a bowl! If we are at a local minimum, we need to inflate the variance to increase our search space. We do this whenever our algorithm does not manage to decrease. However, if after a while we do not get a better result, we assume that this is indeed not a local minimum but rather a global minimum and start deflating the variance. This mechanism of inflation deflation ensures we can handle noisy functions like

Rastrigin or Schwefel 1 or Schwefel 2 functions as defined in the chapter part 6.4 and is similar in the spirit to the combination of local and global (isotropic and anisotropic) search path of traditional CMA-ES

6.3.4 Differences with standard CMA-ES

Since we use a rigorous derivation of the posterior, we have the following features:

- The update of the covariance takes all points. This is different from λ/μ CMA-ES which uses only a subset of the points.
- by design, the update is accurate according to our assumptions as we compute at each step rigorously the posterior.
- The contraction dilatation mechanism is an alternative to the global local search path in standard CMA-ES.
- weights vary across iterations which is also a difference from the main CMA-ES. Weights are proportional to $\exp(\frac{1}{2}X^T\Sigma^{-1}X)$ sorted in decreasing order. Initially, when the variance is large, weights are small. As the variance reduces to zero, weights increase to make the final distribution peak at the obtained minimum.

6.3.5 Full algorithm

The complete Bayesian CMA-ES algorithm is summarized in 5. It iterates until a stopping condition is met. We use multiple stopping conditions. We stop if we have not increased our best result for a given number of iterations. We stop if we have reached the maximum of our iterations. We stop if our variance norm is small. Additional stopping conditions can be incorporated easily.

Algorithm 5 Bayesian update of CMA-ES parameters:

Initialization
Start with a prior distribution Π on μ and Σ
Set retrial to 0
Set f_{min} to max float
while stop criteria not satisfied **do**
  $X \sim \mathcal{N}(\mu, \Sigma)$
  update the parameters of the Gaussian thanks to the posterior law $\Pi(\mu, \Sigma|X)$ following details given in algorithm 4
  Handle dilatation contraction variance for local minima as explained in algorithm 6
  if DilateContractFunc$(X, \overline{\Sigma}_t, X_{min}, f_{min}, \overline{\Sigma}_{t,min}) == 1$ **then**
    return best solution
  end if
end while
return best solution

Last but not least, we have a dilatation contraction mechanism for the variance to handle local minima with multiple levels of contractions and dilatation that is given in the function 6. The overall idea is first to dilate variance if we do not make any progress to increase the search space so

that we are not trapped in a local minimum. Should this not succeed, it means that we are reaching something that looks like the global minimum and we progressively contract the variance. In our implemented algorithm, we take $L_1 = 5, L_2 = 20, L_3 = 30, L_4 = 40, L_5 = 50$ and the dilatation, contraction parameters given by $k_1 = 1.5, k_2 = 0.9, k_3 = 0.7, k_5 = 0.5$ We have also a restart at previous minimum level $L_* = L_2$.

Algorithm 6 Dilatation contraction variance for local minima:

Function DilateContractFunc($X, \overline{\Sigma}_t, X_{min}, f_{min}, \overline{\Sigma}_{t,min}$)
if $f(X) \leq f_{min}$ **then**
  Set $f_{min} = f(X)$
  Memorize current point and its variance:
    • $X_{min} = X$
    • $\overline{\Sigma}_{t,min} = \overline{\Sigma}_t$
  Set retrial = 0
else
  Set retrial += 1
  if retrial == L_* **then**
    Restart at previous best solution:
      • $X = X_{min}$
      • $\overline{\Sigma}_t = \overline{\Sigma}_{t,min}$
  end if
  if L_2 > retrial and retrial > L_1 **then**
    Dilate variance by k_1
  else if L_3 > retrial and retrial $\geq L_2$ **then**
    Contract variance by k_2
  else if L_4 > retrial and retrial $\geq L_3$ **then**
    Contract variance by k_3
  else if L_5 > retrial and retrial $\geq L_4$ **then**
    Contract variance by k_4
  else
    return 1
  end if
  return 0
end if
End Function

6.4 Experimental analysis

6.4.1 Functions examined

We have examined four functions to stress test our algorithm. They are listed in increasing order of complexity for our algorithm and correspond to different types of functions. They are all generalized functions that can be defined for any dimension n. For all, we present the corresponding equation for a variable $x = (x_1, x_2, .., x_n)$ of n dimension. The code is provided in supplementary materials. We have frozen seeds to have *reproducible results*.

Sphere

The most simple function to optimize is the quadratic sphere whose equation is given by (6.14) and represented in figure 6.2. It is also the standard Euclidean norm. It is obviously convex and is a good test of the performance of an optimization method.

$$f(x) = \sum_{i=1}^{n} x_i^2 = \|x\|_2^2 \tag{6.14}$$

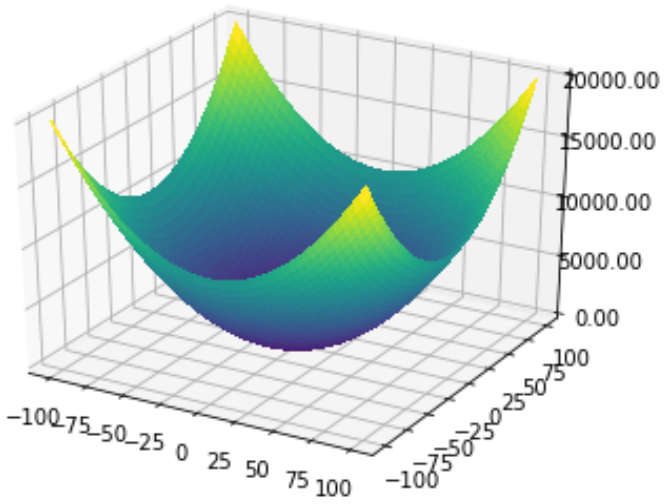


Figure 6.2: A simple convex function: the quadratic norm. Minimum in 0

Schwefel 2 function

A slightly more complicated function is the Schwefel 2 function whose equation is given by (6.15) and represented in figure 6.3. It is a piecewise linear function and validates that the algorithm can cope with non convex functions.

$$f(x) = \sum_{i=1}^{n} | x_i | + \prod_{i=1}^{n} | x_i | \tag{6.15}$$

Rastrigin

The Rastrigin function, first proposed by Rastrigin [1974] and generalized by Mühlenbein et al. [1991], is more difficult compared to the Sphere and the Schwefel 2 function. Its equation is given

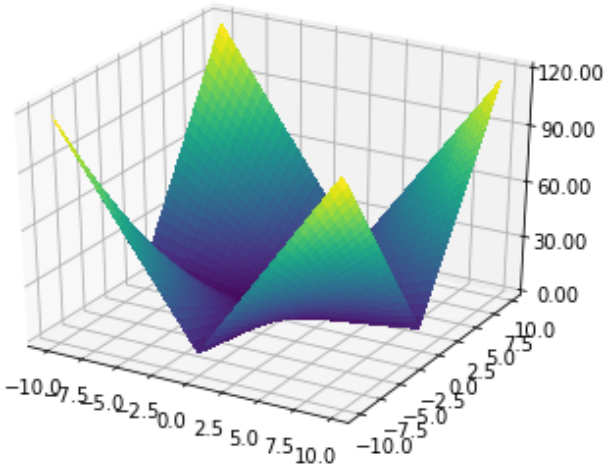


Figure 6.3: Schwefel 2 function: a simple piecewise linear function

by (6.16) and represented in figure 6.4. It is a nonconvex function often used as a performance test problem for optimization algorithms. It is a typical example of non-linear multi modal function. Finding its minimum is considered a good stress test for an optimization algorithm, due to its large search space and its large number of local minima.

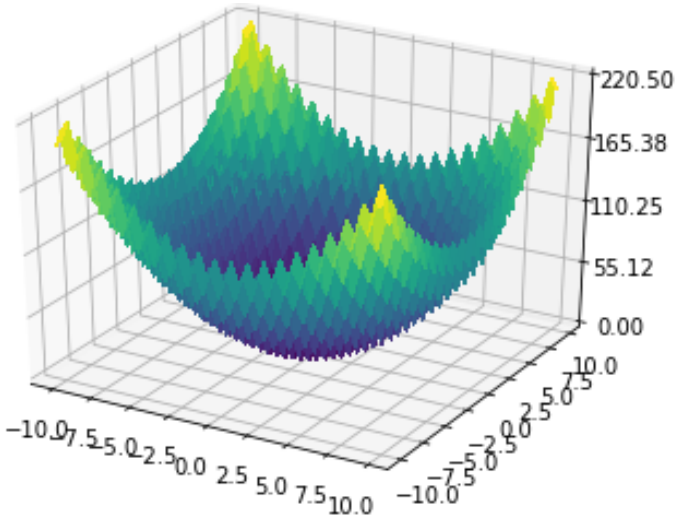


Figure 6.4: Rastrigin function: a non convex function multi-modal and with a large number of local minima

$$f(x) = 10 \times n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) \right] \tag{6.16}$$

Schwefel 1 function

The last function we tested is the Schwefel 1 function whose equation is given by (6.17) and represented in figure 6.5. It is sometimes only defined on $[-500, 500]^n$. The Schwefel 1 function shares similarities with the Rastrigin function. It is continuous, not convex, multi-modal, and with a large number of local minima. The extra difficulty compared to the Rastrigin function, the local minima are more pronounced local bowl making the optimization even harder.

$$f(x) = 418.9829 \times n - \sum_{i=1}^{n} \left[x_i \sin(\sqrt{|x_i|}) \mathbb{1}_{|x_i|<500} + 500 \sin(\sqrt{500}) \mathbb{1}_{|x_i|\geq 500} \right] \tag{6.17}$$

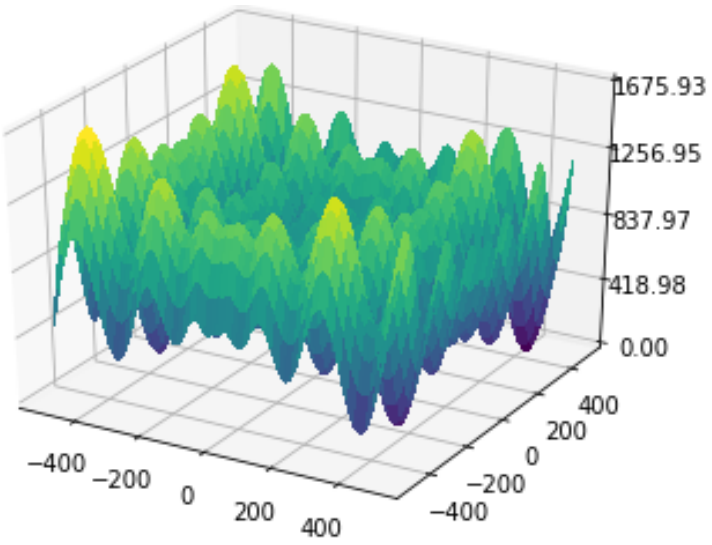


Figure 6.5: Schwefel 1 function: a non convex function multi-modal and with a large number of local pronounced bowls

6.4.2 Convergence

For each of the functions, we compared our method using strategy one entitled *B-CMA-ES S1*: update $\overline{\mu}_t$ and $\overline{\Sigma}_t$ using (6.8) and (6.10) plotted in *orange*, or strategy two *B-CMA-ES S2*: same update but using (6.9) and (6.10), plotted in *blue* and standard CMA-ES as provided by the opensource python package pycma plotted in *green*. We clearly see that strategies one and two are quite similar to standard CMA-ES. The convergence graphics that show the error compared to the minimum are represented:

- for the Sphere function by figure 6.6 (case of a convex function), with initial point $(10, 10)$
- for the Schwefel 2 function in figure 6.7 (case of piecewise linear function), with initial point $(10, 10)$

- for the Rastrigin function in figure 6.8 (case of a non convex function with multiple local minima), with initial point $(10, 10)$
- and for the Schwefel 1 function in figure 6.9 (case of a non convex function with multiple large bowl local minima), with initial point $(400, 400)$

The results are for one test run.

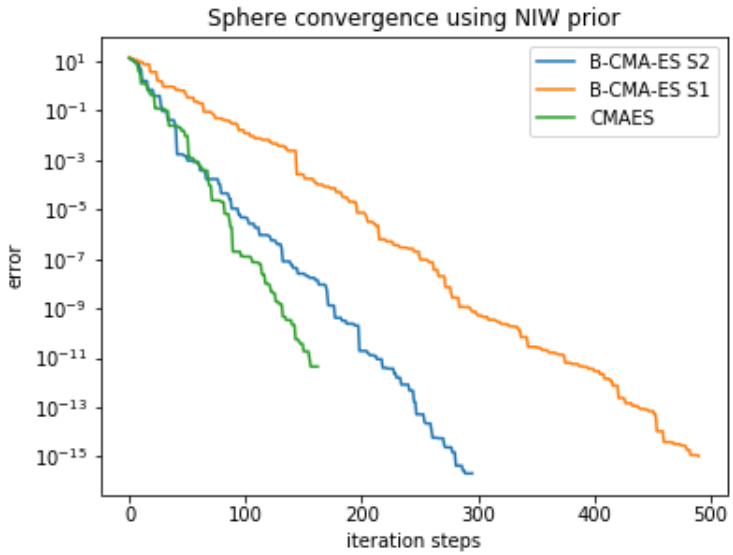


Figure 6.6: Convergence for the sphere function

For the four functions, BCMAES achieves convergence similar to standard CMA-ES. The intuition of this good convergence is that shifting the multi-variate mean by the best candidate seen so far is a good guess to update it at the next run (standard CMA-ES or B-CMA-ES S1).

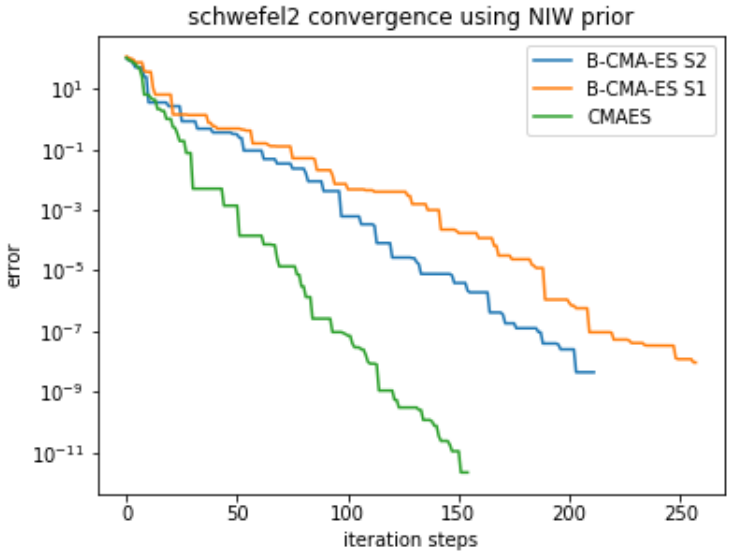


Figure 6.7: Convergence for the Schwefel 2 function

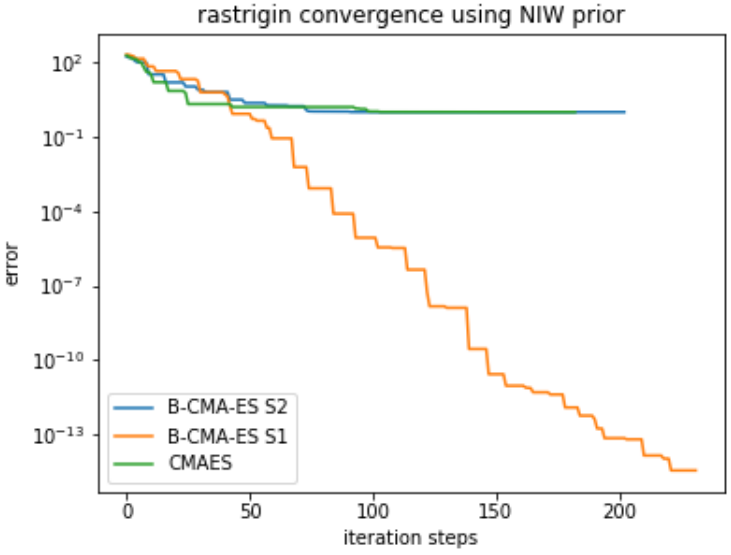


Figure 6.8: Convergence for the Rastrigin function

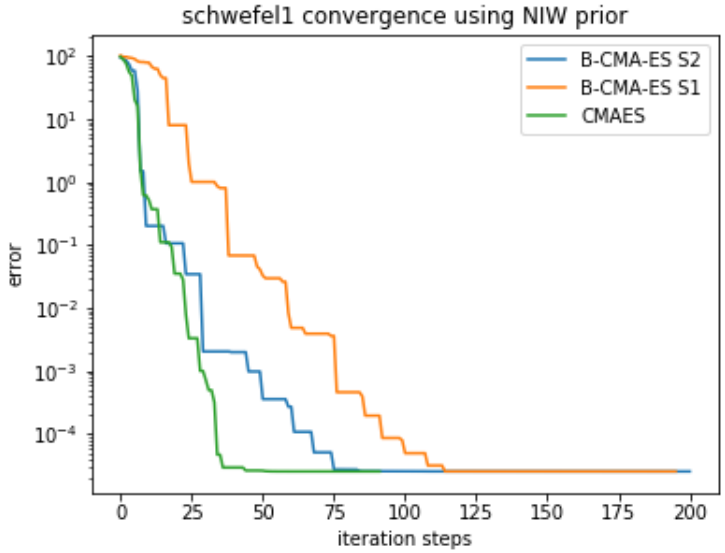


Figure 6.9: Convergence for the Schwefel 1 function

6.5 Appendix

6.5.1 Conjugate priors

Proof. Consider n independent and identically distributed (IID) measurements $\mathcal{X} \triangleq \{\mathbf{x}^j \in \mathbb{R}^d | 1 \leq j \leq n\}$ and assume that these variables have an exponential family density. The likelihood $p(\mathcal{X}|\theta, \lambda)$ writes simply as the product of each individual likelihood:

$$p(\mathcal{X}|\theta,\lambda) = \Big(\prod_{j=1}^{n} h(\mathbf{x}^j)\Big) \exp\Big(\eta(\theta,\lambda)^T \sum_{j=1}^{n} T(x^j) - nA(\eta(\theta,\lambda))\Big). \tag{6.18}$$

If we start with a prior $\pi(\theta)$ of the form $\pi(\theta) \propto \exp(\mathcal{F}(\theta))$ for some function $\mathcal{F}(\cdot)$, its posterior writes:

$$\begin{aligned}\pi(\theta|\mathcal{X}) &\propto p(\mathcal{X}|\theta)\exp(\mathcal{F}(\theta)) \\ &\propto \exp\left(\eta(\theta,\lambda)\cdot\sum_{j=1}^{n} T(x^j) - nA(\eta(\theta,\lambda)) + \mathcal{F}(\theta)\right).\end{aligned} \tag{6.19}$$

It is easy to check that the posterior (6.19) is in the same exponential family as the prior iff $\mathcal{F}(\cdot)$ is in the form:

$$\mathcal{F}(\theta) = \mu_1 \cdot \eta(\theta,\lambda) - \mu_0 A(\eta(\theta,\lambda)) \tag{6.20}$$

for some $\mu \triangleq (\mu_0, \mu_1)$, such that:

$$p(\mathcal{X}|\theta,\lambda) \propto \exp\Big(\Big(\mu_1 + \sum_{j=1}^{n} T(x^j)\Big)^T \eta(\theta,\lambda) - (n+\mu_0)A(\eta(\theta,\lambda))\Big) \tag{6.21}$$

Hence, the conjugate prior for the likelihood (6.18) is parametrized by μ and given by:

$$p(\mathcal{X}|\theta,\lambda) = \frac{1}{Z}\exp\left(\mu_1\cdot\eta(\theta,\lambda) - \mu_0 A(\eta(\theta,\lambda))\right), \tag{6.22}$$

where $Z = \int \exp\left(\mu_1\cdot\eta(\theta,\lambda) - \mu_0 A(\eta(\theta,\lambda))\right)\ \mathrm{d}x$. □

6.5.2 Exact computation of the posterior update for the Normal Inverse Wishart

To make our proof simple, we first start with the one-dimensional case and show that in one dimension it is a normal inverse gamma. We then generalize to the multi-dimensional case.

Lemma 6.11. *The probability density function of a Normal inverse gamma (denoted by NIG) random variable can be expressed as the product of a Normal and an Inverse gamma probability density function.*

Proof. we suppose that $x|\mu, \sigma^2 \sim \mathcal{N}(\mu_0, \sigma^2/v)$. We recall the following definition of conditional probability:

Definition 6.12. *Suppose that events A,B and C are defined on the same probability space, and the event B is such that* $\mathbb{P}(B) > 0$. *We have the following expression:*
$\mathbb{P}(A \cap B|C) = \mathbb{P}(A|B,C)\mathbb{P}(B|C)$.

Applying 6.12, we have:

$$\begin{aligned} p\left(\mu, \sigma^2|\mu_0, v, \alpha, \beta\right) &= p\left(\mu|\sigma^2, \mu_0, v, \alpha, \beta\right) p\left(\sigma^2|\mu_0, v, \alpha, \beta\right) \\ &= p\left(\mu|\sigma^2, \mu_0, v\right) p\left(\sigma^2|\alpha, \beta\right). \end{aligned} \tag{6.23}$$

Using the definition of the Normal inverse gamma law, we end the proof. □

Remark 6.13. If $\left(x, \sigma^2\right) \sim NIG(\mu, \lambda, \alpha, \beta)$, the probability density function is the following:

$$\begin{aligned} f(x, \sigma^2|\mu, \lambda, \alpha, \beta) = \frac{\sqrt{\lambda}}{\sigma\sqrt{2\pi}} \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \\ \exp\left\{-\frac{2\beta + \lambda(x-\mu)^2}{2\sigma^2}\right\}. \end{aligned} \tag{6.24}$$

Proposition 6.14. *The Normal Inverse Gamma NIG* $(\mu_0, v, \alpha, \beta)$ *distribution is a conjugate prior of a normal distribution with unknown mean and variance.*

Proof. the posterior is proportional to the product of the prior and likelihood, then:

$$\begin{aligned} p\left(\mu, \sigma^2|X\right) \propto \frac{\sqrt{v}}{\sqrt{2\pi}} \left(\frac{1}{\sigma^2}\right)^{1/2} \exp\left\{\frac{-v(\mu-\mu_0)^2}{2\sigma^2}\right\} \\ \times \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{\frac{-\beta}{\sigma^2}\right\} \\ \times \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left\{-\frac{\sum_{i=1}^n (x_i-\mu)^2}{2\sigma^2}\right\}. \end{aligned} \tag{6.25}$$

Defining the empirical mean and variance as $\overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$ and $\overline{s} = \frac{1}{n}\sum_{i=1}^n (x_i - \overline{x})^2$, we obtain that $\sum_{i=1}^n (x_i-\mu)^2 = n(\overline{s} + (\overline{x}-\mu)^2)$.

So, the conditional density writes:

$$\begin{aligned} p\left(\mu, \sigma^2|X\right) \propto \sqrt{v}\left(\frac{1}{\sigma^2}\right)^{\alpha+n/2+3/2} \\ \times \exp\left\{-\frac{1}{\sigma^2}\Big[\beta + \frac{1}{2}\Big(v(\mu-\mu_0)^2 + n(\overline{s} + (\overline{x}-\mu)^2)\Big)\Big]\right\}. \end{aligned} \tag{6.26}$$

Besides,

$$\begin{aligned} & v(\mu-\mu_0)^2 + n(\overline{s} + (\overline{x}-\mu)^2) \\ &= v(\mu^2 - 2\mu\mu_0 + \mu_o^2) + n\overline{s} + n(\overline{x}^2 - 2\overline{x}\mu + \mu^2) \\ &= \mu^2(v+n) - 2\mu(v\mu_0 + n\overline{x}) + v\mu_o^2 + n\overline{s} + n\overline{x}^2. \end{aligned} \tag{6.27}$$

Denoting $a = v + n$ and $b = v\mu_0 + n\overline{x}$, we have :

$$\begin{aligned} & \beta + \frac{1}{2}\Big(v(\mu-\mu_0)^2 + n(\overline{s} + (\overline{x}-\mu)^2)\Big) \\ &= \beta + \frac{1}{2}\left(a\mu^2 - 2b\mu + v\mu_o^2 + n\overline{s} + n\overline{x}^2\right) \\ &= \beta + \frac{1}{2}\left(a\left(\mu^2 - \frac{2b}{a}\mu\right) + v\mu_o^2 + n\overline{s} + n\overline{x}^2\right) \\ &= \beta + \frac{1}{2}\left(a\left(\mu - \frac{b}{a}\right)^2 - \frac{b^2}{a} + v\mu_o^2 + n\overline{s} + n\overline{x}^2\right). \end{aligned} \tag{6.28}$$

So we can express the proportional expression of the posterior :

$$p\left(\mu, \sigma^2 | X\right) \propto \left(\frac{1}{\sigma^2}\right)^{\alpha^\star + 3/2} \times \exp\left\{-\frac{2\beta^\star + \lambda^\star(\mu-\mu^\star)^2}{2\sigma^2}\right\},$$

with

- $\alpha^\star = \alpha + \frac{n}{2}$
- $\beta^\star = \beta + \frac{1}{2}\left(\sum_{i=1}^n (x_i - \overline{x})^2 + \frac{nv}{n+v}\frac{(\overline{x}-\mu_0)^2}{2}\right)$
- $\mu^\star = \frac{v\mu_0 + n\overline{x}}{v+n}$
- $\lambda^\star = v + n$

We can identify the terms with the expression of the probability density function given in 6.13 to conclude that the posterior follows a NIG$(\mu^\star, \lambda^\star, \alpha^\star, \beta^\star)$. □

We are now ready to prove the following proposition:

Proposition 6.15. *The Normal Inverse Wishart (denoted by NIW) $(\mu_0, \kappa_0, v_0, \psi)$ distribution is a conjugate prior of a multivariate normal distribution with unknown mean and covariance.*

Proof. we use the fact that the probability density function of a Normal Inverse Wishart random variable can be expressed as the product of a Normal and an Inverse Wishart probability density functions (we use the same reasoning that in 6.11). Besides, the posterior is proportional to the product of the prior and the likelihood.

We first express the probability density function of the multivariate Gaussian random variable in a proper way in order to use it when we write the posterior density function.

$$\begin{aligned} & \textstyle\sum_{i=1}^{n} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \\ = \quad & n (\overline{x} - \mu)^T \Sigma^{-1} (\overline{x} - \mu) + \textstyle\sum_{i=1}^{n} (x_i - \overline{x})^T \Sigma^{-1} (x_i - \overline{x}) . \end{aligned} \tag{6.29}$$

We can inject the previous result and use the properties of the trace function to express the following probability density function of the multivariate Gaussian random variable of parameters μ and Σ. The density writes as:

$$\begin{aligned} & \tfrac{|\Sigma|^{-n/2}}{\sqrt{(2\pi)^{pn}}} \exp \Bigg\{ - \tfrac{n}{2} (\overline{x} - \mu)^T \Sigma^{-1} (\overline{x} - \mu) \\ & \qquad - \tfrac{1}{2} tr \left(\Sigma^{-1} \textstyle\sum_{i=1}^{n} (x_i - \overline{x}) (x_i - \overline{x})^T \right) \Bigg\}. \end{aligned} \tag{6.30}$$

Hence, we can compute explicitly the posterior as follows:

$$\begin{aligned} p\left(\mu, \sigma^2 | X\right) \propto & \frac{\sqrt{\kappa_0}}{\sqrt{(2\pi)^p |\Sigma|}} \exp\left\{ -\frac{\kappa_0}{2} (\mu - \mu_0)^T \Sigma^{-1} (\mu - \mu_0) \right\} \\ & \times \frac{|\psi|^{v/2}}{2^{vp/2} \Gamma_p(v_0/2)} |\Sigma|^{-\frac{v_0+p+1}{2}} exp\left\{ -\frac{1}{2} tr\left(\psi \Sigma^{-1}\right) \right\} \\ & \times |\Sigma|^{-n/2} \exp\Bigg\{ -\frac{n}{2} (\overline{x} - \mu)^T \Sigma^{-1} (\overline{x} - \mu) \\ & \qquad - \frac{1}{2} tr\left(\Sigma^{-1} \sum_{i=1}^{n} (x_i - \overline{x}) (x_i - \overline{x})^T \right) \Bigg\} \end{aligned} \tag{6.31}$$

$$\begin{aligned} \propto |\Sigma|^{-\frac{v_0+p+2+n}{2}} & \exp\Bigg\{ -\frac{\kappa_0}{2} (\mu - \mu_0)^T \Sigma^{-1} (\mu - \mu_0) \\ & - \frac{n}{2} (\overline{x} - \mu)^T \Sigma^{-1} (\overline{x} - \mu) \\ & - \frac{1}{2} tr\left(\Sigma^{-1} \left(\psi + \sum_{i=1}^{n} (x_i - \overline{x}) (x_i - \overline{x})^T \right) \right) \Bigg\}. \end{aligned} \tag{6.32}$$

We organize the terms and find the parameters of our Normal Inverse Wishart random variable $\text{NIW}(\mu_0^\star, \kappa_0^\star, v_0^\star, \psi_0^\star)$.

$$\begin{aligned} & \mu_0^\star = \frac{\kappa_0 \mu_0 + n\overline{x}}{\kappa_0 + n}, \quad \kappa_0^\star = \kappa_0 + n, \quad v_0^\star = v_0 + n \\ & \psi_0^\star = \psi + \sum_{i=1}^{n} (x_i - \overline{x}) (x_i - \overline{x})^T + \frac{\kappa_0 n}{\kappa_0 + n} (\overline{x} - \mu_0) (\overline{x} - \mu_0)^T \end{aligned} \tag{6.33}$$

which are exactly the equations provided in (6.5). □

6.5.3 Weighted combination for the BCMA-ES update

Proof.

$$\begin{aligned}\mathbb{E}_{t+1}[\mu] &= \mu_{t+1} \\ &= \frac{\kappa_t \mu_t + n\overline{\mu}_t}{\kappa_t + n} \\ &= \mathbb{E}_t[\mu] + w_t^{\mu}\left(\overline{\mu}_t - \mathbb{E}_t[\mu]\right)\end{aligned} \tag{6.34}$$

$$\begin{aligned}\mathbb{E}_{t+1}[\Sigma] &= \frac{\psi_{t+1}}{v_{t+1} - n - 1} \\ &= \frac{1}{v_t - 1}\psi_t + \frac{1}{v_t - 1}\overline{\Sigma}_t \\ &\quad + \frac{\kappa_t n}{(\kappa_t + n)(v_t - 1)}(\overline{\mu}_t - \mu_t)(\overline{\mu}_t - \mu_t)^T \\ &= \underbrace{w_t^{\Sigma,1}}_{\text{discount factor}}\mathbb{E}_t[\Sigma] + w_t^{\Sigma,2}\underbrace{(\hat{\mu} - \mathbb{E}_t[\mu])(\hat{\mu} - \mathbb{E}_t[\mu])^T}_{\text{rank one matrix}} \\ &\quad + w_t^{\Sigma,3}\underbrace{\overline{\Sigma}_t}_{\text{rank (n-1) matrix}}\end{aligned} \tag{6.35}$$

where
$$\begin{aligned} w_t^{\mu} &= \frac{n}{\kappa_t + n}, \\ w_t^{\Sigma,1} &= \frac{v_t - n - 1}{v_t - 1}, \\ w_t^{\Sigma,2} &= \frac{\kappa_t n}{(\kappa_t + n)(v_t - 1)}, \\ w_t^{\Sigma,3} &= \frac{1}{v_t - 1} \end{aligned}$$

$\overline{\Sigma}_t$ is a covariance matrix of rank $n - 1$ as we subtract the empirical mean (which removes one degree of freedom). The matrix $(\hat{\mu} - \mathbb{E}_t[\mu])(\hat{\mu} - \mathbb{E}_t[\mu])^T$ is of rank 1 as it is parametrized by the vector $\hat{\mu}$. □

6.6 Summary

In this chapter, we have revisited the CMA-ES algorithm and provided a Bayesian version of it. Taking conjugate priors, we can find optimal updates for the mean and covariance of the multivariate Normal. We have provided the corresponding algorithm which is a new version of CMA-ES. First numerical experiments show this new version is competitive to standard CMA-ES on traditional functions such as Sphere, Schwefel 1, Rastrigin, and Schwefel 2. This faster convergence can be explained on a theoretical side from an optimal update of the prior (thanks to Bayesian update) and the use of the best candidate seen at each simulation to shift the mean of the multi-variate Gaussian likelihood. We envisage further works to benchmark our algorithm to other standard evolutionary algorithms, in particular, to use the COCO platform to provide more meaningful tests and confirm the theoretical intuition of good performance of this new version of CMA-ES, and test the importance of the prior choice.

This chapter has been the subject of a publication Benhamou et al. [2020]. We will conclude the thesis with a presentation of Shapley values and how this has been useful to get an insight into our machine learning models.

7 Using Shapley values to understand the model

7.1 Motivations

Regime changes planning in financial markets is well known to be hard to explain and interpret. Can an asset manager explain clearly the intuition of his regime changes prediction on the equity market ? To answer this question, we consider a gradient boosting decision trees (GBDT) approach to plan regime changes on S&P 500 from a set of 150 technical, fundamental and macroeconomic features. We report an improved accuracy of GBDT over other machine learning (ML) methods on the S&P 500 futures prices. We show that retaining fewer and carefully selected features provides improvements across all ML approaches. Shapley values have recently been introduced from game theory to the field of ML. This approach allows a robust identification of the most important variables planning stock market crises, and of a local explanation of the crisis probability at each date, through a consistent features attribution. We apply this methodology to analyze in detail the March 2020 financial meltdown, for which the model offered a timely out-of-sample prediction. This analysis unveils in particular the contrarian predictive role of the tech equity sector before and after the crash.

7.2 Introduction

From an external point of view, the asset management industry is a well-suited industry to apply machine learning Benhamou et al. [2020a] as large amounts of data are available thanks to the revolution of electronic trading and the methodical collection of data by asset managers or their acquisition from data providers. In addition, machine based decisions can help reduce emotional bias and make rational and systematic investment choices Benhamou et al. [2020b]. A recent major breakthrough has been the capacity to explain the outcome of machine learning decisions and build an intuition on it.
Indeed, regime changes planning in financial markets Benhamou et al. [2021a] is well known to be hard to forecast and explain. The planning of equity crashes, although particularly challenging due to their infrequent nature and the non-stationary features of financial markets, has been the focus of several important works in the past decades. For instance, Sornette and Johansen [2001]

have proposed a deterministic log-periodic model with finite-time explosion to represent equity price bubbles and crashes. In more recent works, Chatzis et al. [2018] and Samitas et al. [2020] have introduced machine-learning approaches to the planning of global equity crises, emphasizing the importance of cross-market contagion effects. Our goal in this work is to use an AI model that is accurate and explainable to solve the multi-agent system environment of financial markets. Hence, we introduce a gradient boosting decision tree (GBDT) approach to predict large falls in the S& P500 equity index, using a large set of technical, fundamental, and macroeconomic features as predictors. Besides illustrating the value of carefully selecting the features and the superior accuracy of GBDT over other ML approaches in some types of small/imbalanced data sets classification problems, our main contribution lies in the explanation of the model predictions at any date. Indeed, from a practitioner's viewpoint, understanding why a model provides certain planning is at least as important as its accuracy. Although the complexity of AI models is often presented as a barrier to a practitioner's understanding of their local planning, the use of SHAP (SHapley Additive exPlanation) values, introduced for the first time by Lundberg and Lee [2017] in machine-learning applications, makes AI models more explainable and transparent. Shapley values are the contributions of each individual feature to the overall crash logit probability. They represent the only set of attributions presenting certain properties of consistency and additivity, as defined by Lundberg. In particular, the most commonly used variable importance measurement methodologies fail to pass the consistency test, which makes it difficult to compare their outputs across different models. Shapley values enlighten both the global understanding and the local explanations of machine learning prediction models, as they may be computed at each point in time.

In our context, this approach first allows us to determine which features most efficiently predict equity crashes (and in which global direction). We infer from a features importance analysis that the S&P500 crash probability is driven by a mix of pro-cyclical and counter-cyclical features. Pro-cyclical features consist either of positive economic/equity market developments that remove the prospect of large equity price drops, or alternatively of negative economic/equity market shocks that portend deadly equity downward spirals. Among these pro-cyclical features, we find the 120-day S&P500 Price/Earnings ratio percent change, the global risk aversion level, the 20-day S&P500 sales percent change, the six-months to one-year US 2 Yrs and 10 Yrs rates evolution, economic surprises indices, and the medium-term industrial metals, European equity indices, and emerging currencies price trends. Conversely, counter-cyclical features can either be positive economic/equity market anticipations predating large equity price corrections, or negative shocks involving a reduced risk of equity downside moves. Important contrarian indicators are the put/call ratio, the six-months S&P500 sales percent change, the 100-day Nasdaq 100 Sharpe ratio, and the U.S. 100-day 10 Yrs real interest change. A second crucial contribution of Shapley values is to help uncover how different features locally contribute to the logit probability at each point in time. We apply this methodology to analyze in detail the unfolding of the events surrounding the March 2020 equity meltdown, for which the model offered a timely prediction out of sample. On January 1, 2020, the crash probability was fairly low, standing at 9.4%. On February 3, we observed a first neat increase in the crash probability (to 27 %), driven by the 100-day Nasdaq Sharpe Ratio contrarian indicator. At the onset of the Covid crash, on March 2, 2020, most pro-cyclical indicators concurred to steeply increase the crash probability (to 61%), as given by figure 7.5, in a way that proved prescient. Interestingly, the Nasdaq 100 index had already started its correction by this date, prompting the tech sector contrarian indicators to switch back in favor of a decreased crash probability. On April 1st, the crash probability plummeted back to 29 %, as the Nasdaq 100 appeared oversold while the Put/Call ratio reflected extremely cautious market

anticipations. Overall, the analysis unveils the role of the tech sector as a powerful contrarian predictor before and after the March 2020 crash.

7.2.1 Related works

Our work can be related to the ever growing field of machine learning applications to financial markets forecasting. Indeed, robust forecasting methods have recently garnered a lot of interest, both from finance scholars and practitioners. This interest can be traced back as early as the late 2000's when machine learning started to pick up. Instead of listing a large number of works, we will refer readers to various works that reviewed the existing literature in chronological order.

In 2009, Atsalakis and Valavanis [2009] surveyed already more than 100 related published articles using neural and neuro-fuzzy techniques derived and applied to forecasting stock markets, or discussing classifications of financial market data and forecasting methods. In 2010, Li and Ma [2010] gave a survey on the application of artificial neural networks in forecasting financial market prices, including exchange rates, stock prices, and financial crisis prediction as well as option pricing. And the stream of machine learning was not only based on neural networks but also genetic and evolutionary algorithms as reviewed in Aguilar-Rivera et al. [2015].

More recently, Xing et al. [2018] reviewed the application of cutting-edge NLP techniques for financial forecasting, using text from financial news or tweets. Rundo et al. [2019] covers the wider topic of machine learning, including deep learning, applications to financial portfolio allocation, and optimization systems. Nti et al. [2019] focused on the use of support vector machine and artificial neural networks to forecast prices and regimes based on fundamental and technical analysis. Later on, Shah et al. [2019] discussed some of the challenges and research opportunities, including issues for algorithmic trading, back testing and live testing on single stocks, and more generally prediction in the financial market. Finally, Sezer et al. [2019] reviewed deep learning as well as other machine learning methods to forecast financial time series. As the hype has been recently mostly on deep learning, it comes as no surprise that most reviewed works relate to this field. One of the only works, to our knowledge, that refers to gradient boosted decision tree applications is Krauss et al. [2017]

Interestingly, Gradient boosting decision trees (GBDT) are almost non-existent in the financial market forecasting literature. As is well-known, GBDT is prone to over-fitting in regression applications. However, they are the method of choice for classification problems as reported by the ML platform Kaggle. In finance, the only space where GBDT has become popular is the credit scoring and retail banking literature. For instance, Brown and Mues [2012] or Marceau et al. [2019] reported that GBDT is the best ML method for this specific task as it can cope with a limited amount of data and very imbalanced classes.

When classifying stock markets into two regimes (a 'normal' one and a 'crisis' one), we are precisely facing very imbalanced classes and a binary classification challenge. In addition, when working with daily observations, we are faced with a ML problem with a limited number of data. These two points can seriously hinder the performance of deep learning algorithms that are well known to be data greedy. Hence, our work investigates whether GBDT can provide a suitable method to identify stock market regimes. In addition, as a byproduct, GBDT provides explicit decision rules (as opposed to deep learning), making it an ideal candidate to investigate regime qualification for stock markets. In this work, we apply our methodology to the US S&P 500 futures prices. In unreported works, we have shown that our approach may easily and successfully

be transposed to other leading stock indices like the Nasdaq, the Eurostoxx, the FTSE, the Nikkei, or the MSCI Emerging futures prices.

Concerning explainable AI (XAI), there has been plenty of research on transparent and interpretable machine learning models, with comprehensive surveys like Adadi and Berrada [2018] or Choo and Liu [2018]. Rosenfeld and Richardson [2019] discusses at length XAI motivations and methods. Furthermore, Liu et al. [2017] regroup XAI into three main categories for understanding, diagnosing, and refining. It also presents applicable examples relating to the prevailing state-of-the-art with upcoming future possibilities. Indeed, explainable systems for machine learning have has applied in multiple fields like plant stress phenotyping Ghosal et al. [2018], heat recycling, fault detection Madhikermi et al. [2019], capsule Gastroenterology Malhi et al. [2019-12] and loan attribution Malhi et al. [2020]. Furthermore, Malhi et al. [2020] uses a combination of LIME and Shapley for understanding and explaining model outputs. Our application is about financial markets which is a more complex multi-agent environment where global explainability is more needed, hence the usage of Shapley to provide intuition and insights to explain and understand model outputs.

7.2.2 Contribution

Our contributions are threefold:

- We specify a valid GBDT methodology to identify stock market regimes, based on a combination of more than 150 features including financial, macro, risk aversion, price, and technical indicators.
- We compare this methodology with other machine learning (ML) methods and report an improved accuracy of GBDT over other ML methods on the S& P 500 futures prices.
- Last but not least, we use Shapley values to provide a global understanding and local explanations of the model at each date, which allows us to analyze in detail the model predictions before and after the March 2020 equity crash.

7.2.3 Why GBDT?

The motivations for Gradient boosting decision trees (GBDT) are multiple:

- GBDT are the most suitable ML methods for small data sets classification problems. In particular, they are known to perform better than their state-of-the-art cousins, Deep Learning methods, for small data sets. As a matter of fact, GBDT methods have been Kagglers' preferred ones and have won multiple challenges.
- GBDT methods are less sensitive to data re-scaling, compared to logistic regression or penalized methods.
- They can cope with imbalanced data sets.
- They allow for very fast training when using leaf-wise tree growth (compared to level-wise tree growth).

7.3 Methodology

In a normal regime, equity markets are rising as investors get rewarded for their risk-taking. This has been referred to as the 'equity risk premium' in the financial economics literature Mehra and Prescott [1985]. However, there are subsequent downturns when financial markets switch to panic mode and start falling sharply. Hence, we can simply assume that there are two equity market regimes:

- a *normal* regime where an asset manager should be positively exposed to benefit from the upward bias in equity markets.
- and a *crisis* regime, where an asset manager should either reduce its equity exposure or even possibly short-sell when permitted.

We define a crisis regime as an occurrence of index return below the historical 5% percentile, computed on the training data set. The 5% is not taken randomly but has been validated historically to provide meaningful levels, indicative of real panic and more importantly forecastable. For instance, in the S&P 500 market, typical levels are returns of -6 to -5 % over a 15-day horizon. To predict whether the coming 15-day return will be below the 5% percentile (hence being classified as in crisis regime), we use more than 150 features described later on. Simply speaking, these 150 features are variables ranging from risk aversion measures to financial metrics indicators like 12-month-forward sales estimates, earning per share, Price/Earnings ratio, economic surprise indices (like the aggregated Citigroup index that compiles major figures like ISM numbers, nonfarm payrolls, unemployment rates, etc).

We only consider two regimes with a specific focus on left-tail events on the returns distribution because we found it easier to characterize extreme returns than to plan outright returns using our set of financial features. In the ML language, our regime detection problem is a pure supervised learning exercise, with a two-regime classification. Hence the probability of being in the normal regime and the one of being in the crisis regime sum to one.

Daily price data are denoted by P_t. The return over a period of d trading days is simply given by the corresponding percentage change over the period: $R_t^d = P_t/P_{t-d} - 1$. The crisis regime is determined by the subset of events where returns are lower or equal to the historical 5% percentile denoted by C. Returns that are below this threshold are labeled "1" while the label value for the normal regime is set to "0". Using traditional binary classification formalism, we denote the training data $X = \{x_i\}_{i=1}^N$ with $x_i \in \mathbb{R}^D$ and their corresponding labels $Y = \{y_i\}_{i=1}^N$ with $y_i \in 0, 1$. The goal is to find the best *classification* function $f^*(x)$ according to the temporal sum of some specific loss function $\mathcal{L}(y_i, f(x_i))$ as follows:

$$f^* = \arg\min_f \sum_{i=1}^{N} \mathcal{L}(y_i, f(x_i))$$

Gradient boosting assumes the function f to take an additive form :

$$f(x) = \sum_{m=1}^{T} f_m(x) \tag{7.1}$$

where T is the number of iterations. The set of weak learners $f_m(x)$ is designed incrementally. At the m-th stage, the newly added function, f_m is chosen to optimize the aggregated loss while

keeping the previously found weak learners $\{f_j\}_{j=1}^{m-1}$ fixed. Each function f_m belongs to a set of parameterized base learners that are modeled as decision trees. Hence, in GBDT, there is an obvious design trade-off between taking a large number of boosted rounds and very simple-based decision trees or a limited number of base learners but of larger size. From our experience, it is better to take small decision trees to avoid over-fitting and an important number of boosted rounds. In this work, we use 500 boosted rounds. The intuition between this choice is to prefer a large crowd of experts that have difficulty memorizing data and should hence avoid over-fitting compared to a small number of strong experts that are represented by large decision trees. Indeed, if these trees go wrong, their failure is not averaged out, as opposed to the first alternative. Typical implementations of GBDT are XGBoost, as presented in Chen and Guestrin [2016], LightGBM as presented Ke et al. [2017], or Catboost as presented Prokhorenkova et al. [2018]. We tested both XGBoost and LightGBM and found a threefold speed for LighGBM compared to XGBoost for similar learning performances. Hence, in the rest of the chapter, we will focus on LightGBM.

For our experiments, we use daily observations of the S&P 500 merged back-adjusted (rolled) futures prices using Homa internal market data. Our daily observations are from 01Jan2003 to 15Jan2021. We split our data into three subsets: a training sample from 01Jan2003 to 31Dec2018, a validation sample used to find the best hyper-parameters from 01Jan2019 to 31Dec2019, and a test sample from 01Jan2020 to 15 Jan2021.

7.3.1 GBDT hyperparamers

The GBDT model contains a high number of hyper-parameters to be specified. From our experience, the following hyper-parameters are very relevant for imbalanced data sets and need to be fine-tuned using evolutionary optimizations as presented in Benhamou et al. [2019c]: min sum hessian in leaf, min gain to split, feature fraction, bagging fraction, and lambda l2. The max depth parameter plays a central role in the use of GBDT. On the S&P 500 futures, we found that very small trees with a max depth of one perform better over time than larger trees. The 5 parameters mentioned above are determined as the best hyperparameters on the validation set.

7.3.2 Features used

The model is fed by more than 150 features to derive a daily 'crash' probability. These data can be grouped into 6 families:

- **Risk aversion metrics** such as equities', currencies', or commodities' implied volatilities, credit spreads, and VIC forward curves.
- **Price indicators** such as returns, Sharpe ratio of major stock markets, distance from the long-term moving average, and equity-bond correlation.
- **Financial metrics** such as sales growth or Price/Earnings ratios forecast 12 months forward.
- **Macroeconomic indicators** such as economic surprises indices by region and globally as given by Citigroup surprise index.
- **Technical indicators** such as market breath or put-call ratio.
- **Rates** such as 10 Yrs and 2 Yrs U.S. rates, or break-even inflation information.

7.3.3 Process of features selection

Using all raw features would add too much noise to our model and would lead to biased decisions. We thus need to select or extract only the most meaningful features. As we can see in figure 7.1, we do so by removing the features in 2 steps:

- Based on gradient boosting trees, we rank the features by importance or contribution.
- We then pay attention to the severity of multicollinearity in an ordinary least squares regression analysis by computing the variance inflation factor (VIF) to remove co-linear features. Considering a linear model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + .. + \beta_n X_n + \epsilon$, the VIF is equal to $\frac{1}{1-R_j^2}$, R_j^2 being the multiple R^2 for the regression of X_j on other covariates. The VIF reflects the presence of collinear factors that increase the variance in the coefficient estimates.

At the end of this 2-part process, we only keep 33% of the initial features.

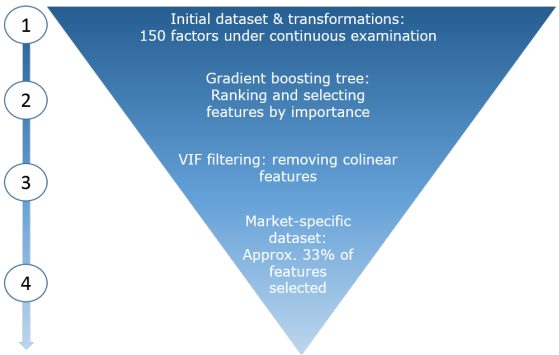


Figure 7.1: 4-steps features selection process

In the next section, we will investigate whether removing correlated features improves the out-of-sample precision of the model.

7.4 Results

7.4.1 Model presentation

Although our work is mostly focused on the GBDT model, we compare it against common ML models, namely:

- a support vector model with a radial basis function kernel, a γ parameter of 2, and a C parameter of 1 (RBF SVM). We use the sklearn implementation. The two hyperparameters γ and C are found on the validation set.
- a Random Forest (RF) model, whose max depth is set to 1 and boosted rounds are set to 500. We purposely tune the RF model similarly to our GBDT model in order to benefit from the above-mentioned error averaging feature. We found this parameter combination to perform well for annual validation data sets ranging from the year 2015 onward on the S&P 500 market. We note that a max depth of 1 does not allow for interaction effects between features.

- a first deep learning model referred to in our experiment as Deep FC (for fully connected layers), which is naively built with three fully connected layers (64, 32, and one for the final layer) with a drop out in of 5 % between and Relu activation, whose implementation details rely on TensorFlow keras 2.0.
- a second more advanced deep learning model consisting of two layers referred to in our experiment as deep LSTM: a 64 nodes LSTM layer followed by a 5% dropout followed by a 32 nodes dense layer followed by a dense layer with a single node and a sigmoïd activation.

For both deep learning models, we use a standard Adam optimizer whose benefit is to combine adaptive gradient descent with root mean square propagation Kingma and Ba [2014].

We train each model using either the full set of features or only the filtered ones, as described in 7.1. Hence, for each model, we add a suffix ' raw' or ' FS' to specify if the model is trained on the full set of features or after features selections. We provide the performance of these models according to different metrics, namely accuracy, precision, recall, f1-score, AUC and AUC-pr in tables 7.1 and 7.2. The GBDT with features selection is superior according to all metrics and outperforms in particular the deep learning model based on LSTM, confirming the consensus reached in the ML community as regards classification problems in small and imbalanced data sets.

Table 7.1: Model comparison

Model	accuracy	precision	recall
GBDT FS	**0.89**	**0.55**	**0.55**
Deep LSTM FS	0.87	0.06	0.02
RBF SVM FS	0.87	0.03	0.07
Random Forest FS	0.87	0.03	0.07
Deep FC FS	0.87	0.01	0.02
Deep LSTM Raw	0.84	0.37	0.33
RBF SVM Raw	0.87	0.02	0.01
Random Forest Raw	0.86	0.30	0.09
GBDT Raw	0.86	0.20	0.03
Deep FC Raw	0.85	0.07	0.05

7.4.2 AUC performance

Figure 7.2 provides the ROC Curve for the two best performing models, the GBDT and the Deep learning LSTM model with features selection. ROC curves enable us to visualize and analyze the relationship between precision and recall and to investigate whether the model makes more type I or type II errors when identifying market regimes. The receiver operating characteristic (ROC) curve plots the true positive rate (sensitivity) on the vertical axis against the false positive rate (1 - specificity, fall-out) on the horizontal axis for all possible threshold values. The two curves are well above the *blind guess* benchmark that is represented by the dotted red line. This effectively demonstrates that these two models have some predictability power, although being far from a perfect score that would be represented by a half square. Furthermore, the area under the GBDT curve with features selection is 0.83, to be compared with 0.74, one of the second best

Table 7.2: Model comparison

Model	f1-score	auc	auc-pr
GBDT FS	**0.55**	**0.83**	**0.58**
Deep LSTM FS	0.05	0.74	0.56
RBF SVM FS	0.06	0.50	0.56
Random Forest FS	0.04	0.54	0.56
Deep FC FS	0.04	0.50	0.56
Deep LSTM Raw	0.35	0.63	0.39
RBF SVM Raw	0.05	0.50	0.36
Random Forest Raw	0.14	0.53	0.25
GBDT Raw	0.05	0.51	0.18
Deep FC Raw	0.02	0.49	0.06

model (deep LSTM), also with Features selection. Its curve, in blue, is mostly over the one of the second-best models (deep LSTM), in red, which indicates that in most situations, GBDT model performs better than the deep LSTM model.

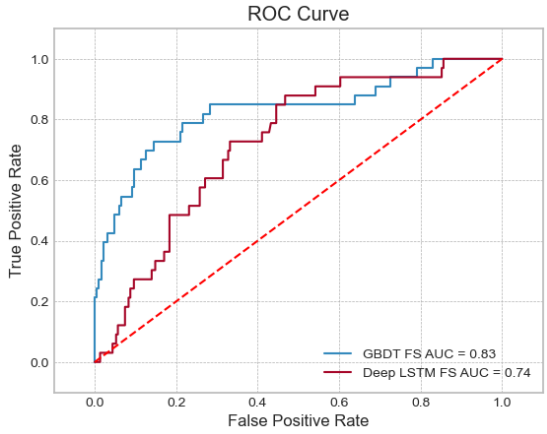


Figure 7.2: ROC Curve of the two best models

7.5 Understanding the model

7.5.1 Shapley values

Building on the work of Lundberg and Lee [2017], we use Shapley values to represent the contribution of each feature to the crisis probability. SHAP (SHapley Additive exPlanation) values explain the output of a function f as a sum of the effects of each feature. It assigns an importance value to each feature that represents the effect on the model planning of including that feature. To compute this effect, a model $f_{S\cup\{i\}}$ is trained with that feature present, and another model f_S is trained with the feature withheld. This method hence requires retraining the model on all feature subsets $S \subseteq M \setminus \{i\}$, where M is the set of all features. Then, predictions from the two models are compared through the difference $f_{S\cup\{i\}}(x_{S\cup\{i\}})$ - $f_S(x_S)$, where x_S represents the values of the input

features in the set S. Since the effect of withholding a feature depends on other features in the model, the preceding differences are computed on all possible differences $f_{S\cup\{i\}}(x_{S\cup\{i\}})$ - $f_S(x_S)$, for all possible subsets $S \subseteq M \setminus \{i\}$. Shapley values are then constructed as a weighted average of all these differences, as follows:
The Shapley value Φ_i attributed to feature i is defined as:

$$\Phi_i = \sum_{S\subseteq M\setminus\{i\}} \frac{|S|!\,(|M|-|S|-1)!}{|M|!}\left(f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right)$$

where $|A|$ refers to the cardinal of the set A, M is the complete set of features, S is the subset of features used, and x_S represents the values of the input features in the set S. Proofs from game theory show that Shapley values are the only possible consistent approach such that the sum of the feature attributions is equal to the output of the function we are to explain. Compared to LIME as presented in Ribeiro et al. [2016], Shap has the advantage of consistency and focuses on global interpretability versus local for LIME.

The exact computation of SHAP values is challenging. In practice, assuming features' independence, we approximate $f_S(x_S)$ by the Shapley 'sampling value', i.e. the conditional mean of the global model prediction $\hat{f}(X)$ (calibrated on the complete set of features), marginalizing over the values x_C of features that are not included in set S:

$$f_S(x_S) = \mathbb{E}[\hat{f}(X)|x_S] \approx \int \hat{f}(x_S, x_C)p(x_C)dx_C$$

In our application, with max depth of 1, interaction effects between features are discarded, which allows computing Shapley values trivially from equation (7.1).

7.5.2 Shapley interpretation

We can rank the Shapley values by order of magnitude importance, defined as the average absolute Shapley value over the training set of the model: this is essentially the average impact on model output when a feature becomes "hidden" from the model. Furthermore, the correlation of a feature to its Shapley value provides insight into the effect of this feature on the probability of a stock market crash. Figure 7.3 represents the joint behavior of Shapley and features values to better grasp their non-linear dependencies.

Concerning figure 7.3, we observe that the most significant feature is the 250 days percent change in S&P 500 Price Earnings ratio (using the forward 1 Yr Earnings of the index as provided by Bloomberg). This reflects the presence of persistent cycles during which market participants' bullish anticipations regarding future earnings growths and market valuations translate into reduced downside risk for equity prices.

By the same token, a positive (resp. negative) 250-day change in the US 2 Yrs yield characterizes a regime of growth (resp. recession) in equities. A positive change in the Bloomberg Base Metals index is associated with a reduced crash probability. The same reasoning applies to the FX Emerging Basket, the S&P Sales evolution, the Euro Stoxx distance to its 200-day moving average and the EU Economic Surprise Index. Similarly, a higher (resp. lower) Risk Aversion implies a higher (resp. lower) crash probability and the same relationship is observed for the realized 10-day S&P 500 volatility.

Interestingly, the model identifies the Put/Call ratio as a powerful contrarian indicator. Indeed, a persistently low level of the Put/Call ratio (as reflected by a low 20-day moving average) reflects over-optimistic expectations and therefore an under-hedged market. Last but not least, the Nasdaq 100 is identified as a contrarian indicator: the higher the Nasdaq 20-day percent change and the higher the Nasdaq 100-day and 250-day Sharpe Ratios, the higher the crash probability. More generally, foreign markets are used pro-cyclically (Euro Stoxx, BCOM Industrials, FX emerging) whereas most domestic price indicators are used counter-cyclically (Nasdaq 100, S&P 500). This is an example where we can see some strong added value from the machine learning approach over a human approach, as the former combines contrarian and trend-following signals while the latter is generally biased towards one type of signal.

7.5.3 Joint features and Shapley Values Distribution

Because some of the features have a strongly non-linear relation to the crisis probability, we also display in figure 7.3 the joint behavior of features and Shapley values at each point in time. The y-axis reports the Shapley values, i.e. the feature contributions to the model output in log-odds (we recall that the GDBT model has a logistic loss) while the color of the dot represents the value of that feature at each point in time. This representation uncovers the non-linearities in the relationship between the Shapley values and the features.

For instance, a large 250-day increase in the P/E ratio (in red color) has a negative impact on the crash probability, everything else equal. The same type of dependency is observed for the change in US 10 Yrs and 2 Yrs yields: the higher (resp. lower) the change in yield, the lower (resp. higher) the crash probability.

However, the dependency of the Shapley value to the 120-day BCOM Industrial Metals Sharpe ratio is non-linear. Elevated Sharpe ratios portend a lower crash probability while the relation vanishes for low Sharpe ratios. An ambiguous dependency is also observed for the 100-day Emerging FX Sharpe ratio, which explains its muted correlation to the Shapley value. This observation is all the more striking as this feature is identified as important in terms of its global absolute impact on the model output. By the same token, the 20-day percent change in S&P 500 Sales does not display a linear relationship to the crash probability. First of all, mostly elevated values of the change in sales are used by the model. Second, large increases in S&P 500 sales are most of the time associated with a drop in the crash probability, but not in every instance.

The impact of the distance of the Euro Stoxx 50 to its 200-day moving average is mostly unambiguous. Elevated levels in the feature's distribution generally (but not systematically) involve a decrease in the crash probability, whereas low levels of this feature portend an increased likelihood of crisis. The 20-day Moving Average of the Put/Call Ratio intervenes as a linear contrarian predictor of the crash probability.

The 20-day percent change in the Nasdaq 100 price is confirmed as a contrarian indicator. As illustrated in Figure 7.3, the impact of the 20-day Nasdaq 100 returns is non-linear, as negative returns may predict strongly reduced crash probabilities, while positive returns result in a more moderate increase in the crash probability. Conversely, the 20-day Euro Stoxx returns have a pro-cyclical linear impact on the crash probability, as confirmed by figure 7.3. As previously stated, the GBDT model uses non-US markets in a pro-cyclical way and U.S. markets in a contrarian manner.

7.5.4 Local explanation of the Covid March 2020 meltdown

Not only can Shapley values provide a global interpretation of features' impacts, as described in section 7.5.2 and in figure 7.3, but they can also convey local explanations at every single date.

Figure 7.4 provides the Shapley values for the model in February, 2020. At this date, the model was still positive on the S&P 500 as the crash probability was fairly low, standing at 9.4%. The 6% 120-day increase in the P/E ratio, the low-risk aversion level, reflecting ample liquidity conditions, and the positive EU Economic Surprise index all concurred to produce a low crash probability. However, the decline in the US LIBOR rate, which conveyed gloomy projections on the U.S. economy, and the elevated Put/Call ratio, reflecting excessive speculative behavior, both contributed positively to the crash probability. On February 3, we observed a first steep increase in the crash probability, driven by the 100-day Nasdaq Sharpe Ratio contrarian indicator. At the onset of the Covid crash, on March 2, 2020, the crash probability dramatically increased on the back of deteriorating industrial metals dynamics, falling Euro Stoxx and FTSE prices, negative EU economic surprises, and decreasing S&P 500 P/E, which caused the model to identify a downturn in the equities' cycle. This prediction eventually proved prescient. Interestingly, the Nasdaq 100 index had already started its correction by this date, prompting the tech sector contrarian indicators to switch back in favor of a decreased crash probability.

We can do the same exercise on April 1, 2020. The crash probability plummeted as contrarian indicators started to balance pro-cyclical indicators: the Nasdaq 100 appeared oversold while the Put/Call ratio reflected extremely cautious market anticipations. During several months, the crash probability stabilized between 20% and 30% until the start of July, which showed a noticeable decline of the probability to 11.2%. The P/E cycle started improving and the momentum signals on base metals and other equities started switching sides. Although the crash probability fluctuated, it remained contained throughout the rest of the year.

Last but not least, if we look at Shapley value at the beginning of December 2020, we can draw further conclusions. At the turn of the year, most signals were positive on the back of improving industrial metals, recovering European equity market dynamics, and improving liquidity conditions (reflected by a falling dollar index and a low Risk Aversion). Although this positive picture is balanced by falling LIBOR rates and various small contributors, the features' vote sharply leans in favor of the bullish side.

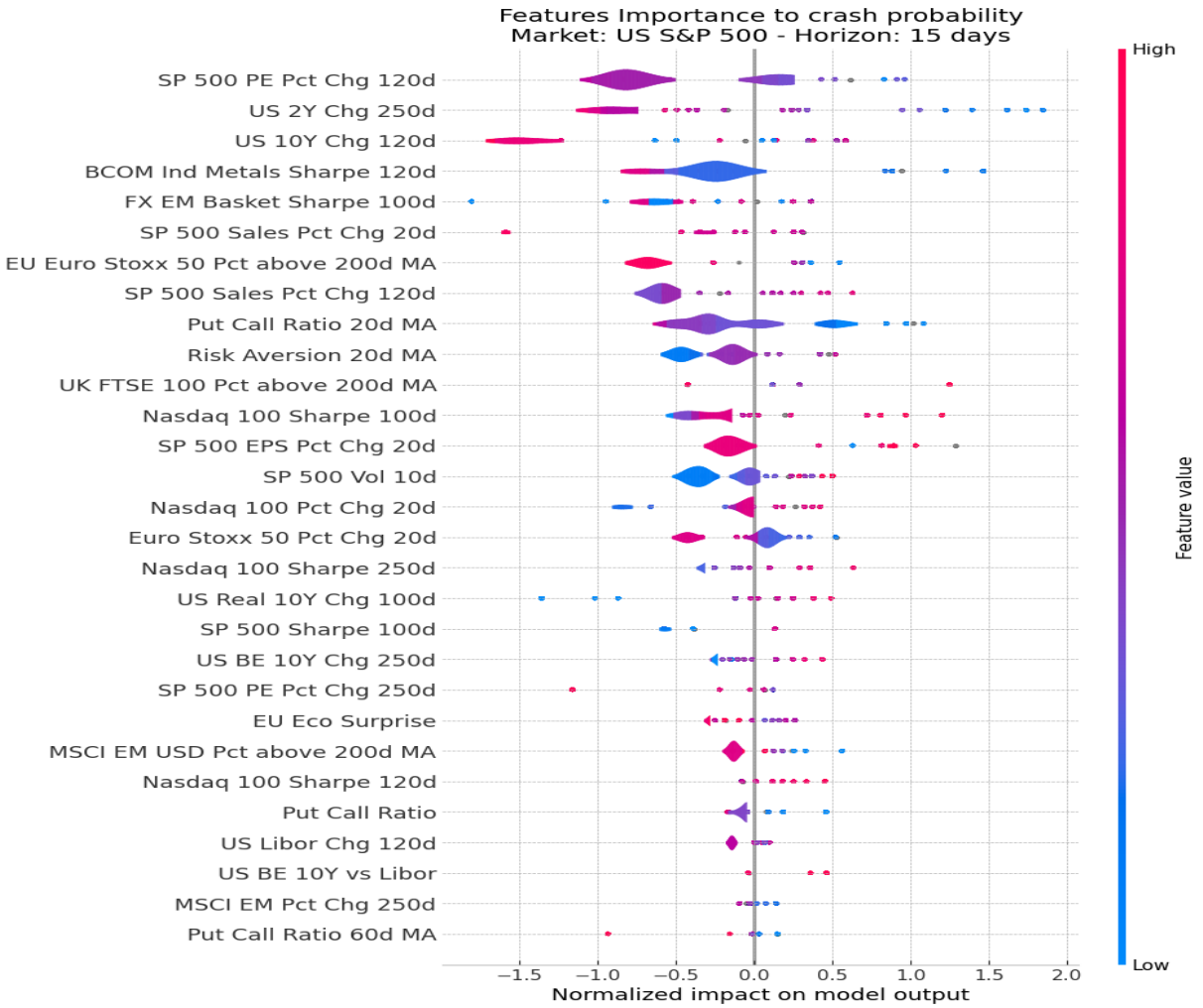


Figure 7.3: Marginal contribution of features with full distribution

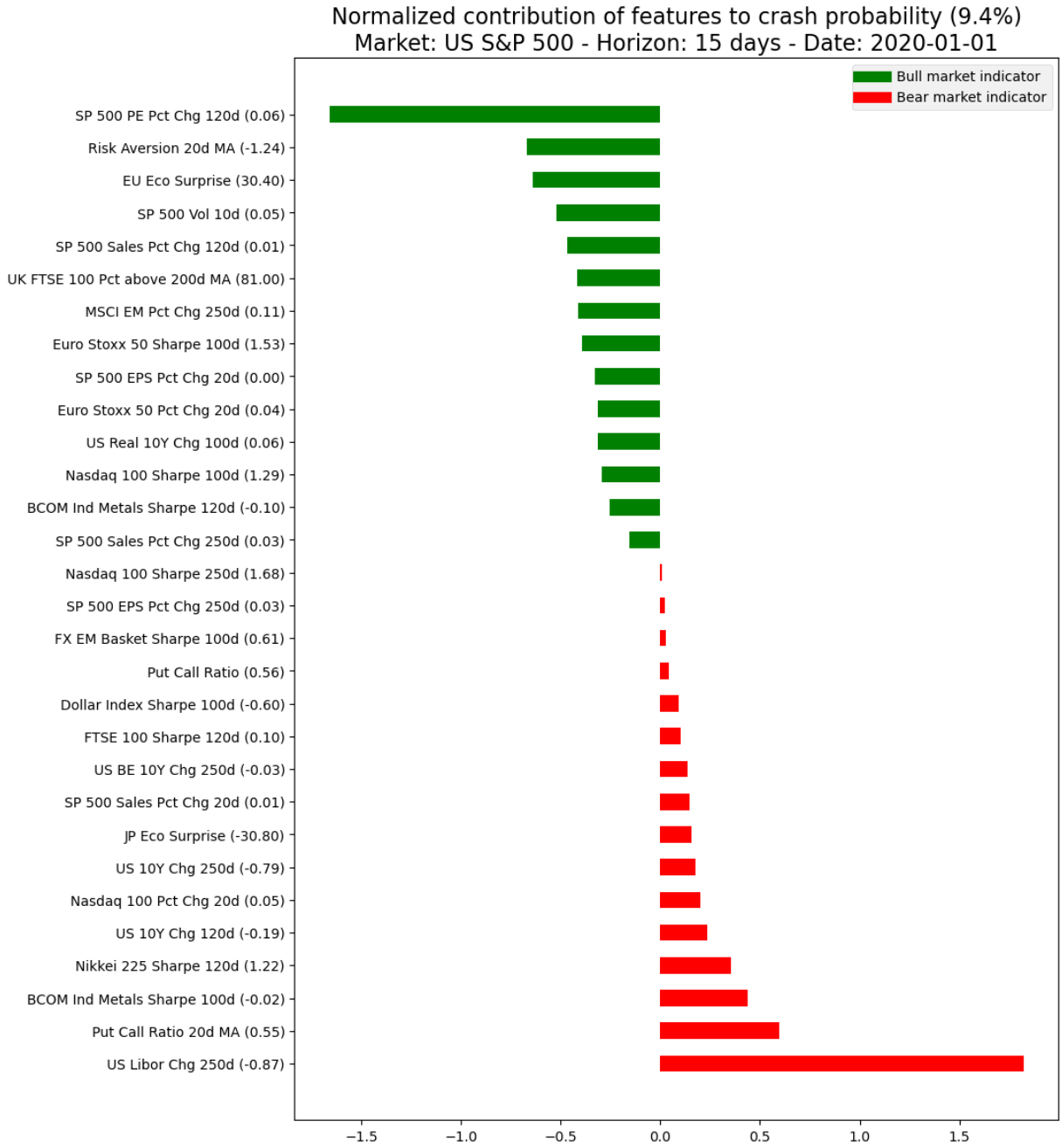


Figure 7.4: Shapley values for 2020-01-01

Figure 7.5: Shapley values for 2020-03-02

7.6 Summary

In this final chapter of the thesis, we have shown how the GBDT method may classify financial markets into normal and crisis regimes, using 150 technical and fundamental features. When applied to the S&P 500, the method yields a high out-of-sample AUC score, which suggests that the machine is able to efficiently learn from previous crises. Our approach also displays an improved accuracy compared to other ML methods, confirming the relevance of GBDT in solving highly imbalanced classification problems with a limited number of observations. AI models complexity is often a barrier to a practitioner's understanding of their local predictions. Yet, from the practitioner's viewpoint, understanding why a model provides a certain prediction is at least as important as the accuracy of this prediction. Shapley values allow for a global understanding of the model behavior and for a local explanation of each feature's contribution to the crash probability at each observation date. This framework shed light on the unfolding of the model predictions during the events that surrounded the March 2020 equity meltdown. In particular, we unveiled the role of the tech equity sector as a powerful contrarian predictor during this episode.

This chapter concludes the thesis. The ideas developed in this chapter have been featured in a publication to a conference, see Ohana et al. [2021].

PART IV

Conclusion

Conclusion and Future Works

In this thesis, we studied the problem of using reinforcement learning in quantitative asset management from different angles using a series of theoretical and practical tools. We have tried to analyze the problem using both a model-based and a model-free approach. In the model-based approach, we examine two different problems. We first address the problem of selecting trades using a supervised learning method. We identify that one of the main challenges is to select the features. We validate experimentally that the supervised learning approach allows improving the performance of the initial trading model by removing trades assumed to be non-profitable. We then examine a second model-based problem where the objective is to select a model whose allocations are proportional to the inverse of the predicted volatility of the strategy. In the second part, we look at the problem of a model-free approach for doing portfolio allocation. The problem to solve is to find a mapping between states that represent the status of financial markets strategies and the allocation of the strategies such that this mapping function maximizes a reward function. The reward function can be customized and can range from a naive net performance to a function mitigating risk and reward like Sharpe ratio or Treynor ratio. We also proved that the deep reinforcement learning method generalizes traditional portfolio methods by extending the initial optimization program to a more general setting. Analyzing experiments, we validate the deep reinforcement learning method provides improvement over traditional portfolio methods.

Overall, our work advocates for the combination of model-based and model-free approach as both solutions work.

Key findings

We can summarize our findings as follows

- When using supervised learning to follow the decision of a model, features selection is critical and conditions most of the failure or success of the machine learning overlay. Hence methods for selecting features are critical. This is a challenging task as features that are relevant for one period may not be relevant in the future. This motivates for walk-forward

analysis that keeps the chronological order of the time series while providing a way to test for multiple periods and test for the robustness of new data and new training of the model.

- When using deep reinforcement learning alone in a model-free setting, we can generalize and improve traditional portfolio allocation methods by mapping the allocation decision to more predictive and forward looking variables than simply the mean and the variance of a portfolio. The optimization problem is non trivial and requires more computing power. However, it leads to more efficient portfolio allocation, when measured in terms of resulting Sharpe ratio or maximum draw-downs, with the former increased and the latter reduced. We have experimented that contextual data are efficient in increasing the accuracy of the Deep RL model. This result which is rather intuitive from a human point of view emphasizes the need of extending the financial states of traditional DRL models often restricted to the time series of strategies returns and variances to other types of data to provide a way for the machine learning model to learn co-dependance between different variables.
- In contrast to traditional portfolio methods, deep reinforcement learning does not play the game of strong diversification and tries in contrast to guess the best performing asset. It is therefore critical to mitigate this natural tendency to concentrate allocation by reward function that offset the risk of over concentration.
- In this thesis, we have highlighted the connections between supervised and reinforcement learning and paved the way for combining these two types of learning. We have also shown that the optimization of hyper-parameters done by Black box optimization methods like CMAES can be indeed reformulated as a non standard Bayesian optimization with appropriate conjugate prior to the Gaussian distribution like normal Wishart or inverse-Wishart distribution.

Moreover, we conjecture that machine learning will become one of the most used tools in asset management to complement or disrupt traditional quantitative methods. The interest of the machine learning approach is that it can not only help in selecting decisions from traditional quantitative models, creating the so called augmented asset manager in a model-based approach or in contrast, in a model-free approach, help finding investment decisions directly from raw inputs and data without any preconceived rules or bias. In particular, the deep reinforcement learning paradigm opens new doors compared to traditional portfolio allocation methods as:

- it naturally links market states to the portfolio allocation, making it able to adapt to changing environment
- it can cope with various objectives as the reward function can encompass agent goals
- it is more forward looking than the traditional method as it optimizes a policy that is computed in the future compared to states
- it does not suffer from any pre-conceived risk or model bias

Future directions

Our work opens up many research avenues in the long term.

The first perspective is to see if we can combine different types of learning in our approach, hence having auxiliary supervised learning tasks that are in turn inputs of a second step reinforcement learning approach. This question of combining different approaches or models in machine learning is something largely emphasized in model stacking in traditional machine learning challenges like in Kaggle competition. It is largely known from an experimental point of view that combining different and uncorrelated models often leads to substantial improvement and robustness.

A second perspective is to continue improving features selection as machine learning relies heavily on selected data. Too much data kills the data. Hence being able to distinguish predictive from noisy data remains a strong motivation for machine learning approach.

A third perspective is how to leverage more Shapley values to do features selections. Shapley values are a great tool to provide the marginal impact of a feature on the final model output. In addition, it provides a global overview of the impact of these variables in conjunction with the other features of the data set. This helps understanding and building an intuition of the motivation and sensitivity of a model to the change of features. We could therefore use this knowledge to do feature selection based on some improved feature importance as Shapley values give a way to quantify the impact of a feature and hence a capacity to remove non impactful features.

PART V

Appendix: Publications and bibliography

A

Publications

In this Appendix, we enumerate publications made during this PhD thesis.

1. **MIDAS 2021: best paper award**
   E Benhamou, D Saltiel, S Tabachnik, C Bourdeix and F Chareyron
   Adaptive Supervised Learning for Financial Markets Volatility Targeting Models
   http://midas.portici.enea.it/midas2021

2. **ICPR 2021: full paper**
   E Benhamou, D Saltiel, JJ Ohana, J Atif
   Detecting and adapting to crisis pattern with context based Deep RL
   https://www.micc.unifi.it/icpr2020/

3. **ESANN 2020: full paper**
   E Benhamou, D Saltiel,
   Similarities between policy gradient methods in RL and SL
   https://www.esann.org/sites/default/files/proceedings/2020/ES2020-34.pdf

4. **Risk Forum 2019: full paper**
   D Saltiel, E Benhamou
   Trade Selection with Supervised Learning and OCA
   https://www.risks-forum.org/RF2019.html

5. **EXTRAAMAS 2021: workshop**
   JJ Ohana, S Ohana, E Benhamou, D Saltiel and B Guez
   Explainable AI (XAI) models applied to the multi agents environment of financial markets
   https://extraamas.ehealth.hevs.ch/archive.html#presentations-2021

6. **ALA 2021: AAMAS workshop**
   E Benhamou, D Saltiel, S Tabachnik, S-K Wong and F Chareyron
   Adaptive learning for financial markets mixing model-based and model-free RL
   https://ala2021.vub.ac.be/

7. **AAAI 2021: KDF workshop**

E Benhamou, D Saltiel, S Ungari, J. Atif
Knowledge discovery with Deep RL for selecting financial hedges
https://aaai.org/Conferences/AAAI-21/ws21/

8. **ICAPS 2020: workshop Finplan**
E Benhamou, D Saltiel, S Ungari, A Mukhopadhyay
Time your hedge with Deep RL
https://icaps20subpages.icaps-conference.org/workshops/finplan/

9. **ICAPS 2020: PRL workshop**
E Benhamou, D Saltiel, S Ungari, A Mukhopadhyay
Bridging the gap between Markowitz planning and deep reinforcement learning
https://icaps20subpages.icaps-conference.org/workshops/prl/

10. **MIDAS 2020: workshop**
D Saltiel, E Benhamou, R Laraki, and J Atif
Trade Selection with Supervised Learning and OCA
http://midas.portici.enea.it/midas2020

11. **GECCO 2020: poster**
E Benhamou, D Saltiel, S Verel
Bayesian CMA-ES: a new approach
https://dl.acm.org/doi/10.1145/3377929.3389913

12. **ECML PKDD 2020: demo paper**
E Benhamou, D Saltiel, JJ Ohana, R. Laraki, J Atif
DRLPS: Deep Reinforcement Learning for Portfolio Selection
https://ecmlpkdd2020.net/programme/accepted/#allTab

Bibliography

A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

R. Aguilar-Rivera, M. Valenzuela-Rendón, and J. Rodríguez-Ortiz. Genetic algorithms and darwinian approaches in financial applications: A survey. *Expert Systems with Applications*, 42 (21):7684–7697, 2015. ISSN 0957-4174.

Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Bidirectional relation between cma evolution strategies and natural evolution strategies. *PPSN*, XI(1):154–163, 2010.

Y. Akimoto, A. Auger, and N. Hansen. Continuous optimization and CMA-ES. *GECCO 2015, Madrid, Spain*, 1:313–344, 2015.

Y. Akimoto, A. Auger, and N. Hansen. CMA-ES and advanced adaptation mechanisms. *GECCO, Denver*, 2016:533–562, 2016.

H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69:279–305, 1994.

K. Astrom. Optimal control of markov processes with incomplete state-information ii. the convexity of the lossfunction. *Journal of Mathematical Analysis and Applications*, 26(2):403–406, 1969.

G. S. Atsalakis and K. P. Valavanis. Surveying stock market forecasting techniques – part ii: Soft computing methods. *Expert Systems with Applications*, 36(3, Part 2):5932–5941, 2009.

A. Auger and N. Hansen. Benchmarking the (1+1)-CMA-ES on the BBOB-2009 noisy testbed. *Companion Material*, GECCO 2009:2467–2472, 2009.

A. Auger and N. Hansen. Tutorial CMA-ES: evolution strategies and covariance matrix adaptation. *Companion Material Proceedings*, 2012(12):827–848, 2012.

A. Auger, M. Schoenauer, and N. Vanhaecke. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. *PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, 2004(2004):182–191, 2004.

W. Bao and X. yang Liu. Multi-agent deep reinforcement learning for liquidation strategy analysis, 2019.

E. Benhamou. Trend without hiccups: a kalman filter approach. *ssrn.1808.03297*, 2018.

E. Benhamou. Connecting sharpe ratio and student t-statistic, and beyond. *ArXiv*, 2019.

E. Benhamou and B. Guez. Incremental sharpe and other performance ratios. *Journal of Statistical and Econometric Methods*, 2018, 2018.

E. Benhamou and D. Saltiel. Similarities between policy gradient methods in reinforcement and supervised learning. In *ESANN proceedings*, 2020.

E. Benhamou, B. Guez, and N. Paris1. Omega and sharpe ratio. *ArXiv*, 2019a.

E. Benhamou, D. Saltiel, B. Guez, and N. Paris. Testing sharpe ratio: luck or skill? *ArXiv*, 2019b.

E. Benhamou, D. Saltiel, S. Vérel, and F. Teytaud. BCMA-ES: A bayesian approach to CMA-ES. *CoRR*, abs/1904.01401, 2019c.

E. Benhamou, D. Saltiel, S. Ungari, and A. Mukhopadhyay. Bridging the gap between markowitz planning and deep reinforcement learning. In *ICAPS proceedings*, 2020a.

E. Benhamou, D. Saltiel, S. Ungari, and A. Mukhopadhyay. Time your hedge with deep reinforcement learning. In *ICAPS proceedings*, 2020b.

E. Benhamou, D. Saltiel, S. Ungari, and A. Mukhopadhyay. Aamdrl: Augmented asset management with deep reinforcement learning. *arXiv*, 2020c.

E. Benhamou, D. Saltiel, and S. Verel. Bayesian CMA-ES: a new approach. In *GECCO proceedings*, 2020.

E. Benhamou, D. Saltiel, J.-J. Ohana, and J. Atif. Detecting and adapting to crisis pattern with context based deep reinforcement learning. In *ICPR 2021 proceedings*, 2021a.

E. Benhamou, D. Saltiel, J. J. Ohana, J. Atif, and R. Laraki. Deep reinforcement learning (drl) for portfolio allocation. In Y. Dong, G. Ifrim, D. Mladenić, C. Saunders, and S. Van Hoecke, editors, *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*, pages 527–531, Cham, 2021b. Springer International Publishing.

E. Benhamou, D. Saltiel, S. Tabachnik, C. Bourdeix, and F. Chareyron. Adaptive supervised learning for financial markets volatility targeting models. In *MIDAS: best paper award*, 2021c.

E. Benhamou, D. Saltiel, S. Tabachnik, S. K. Wong, and F. Chareyron. Adaptive learning for financial markets mixing model-based and model-free rl for volatility targeting. In *AAAMAS: ALA*. AAAI Press, 2021d.

E. Benhamou, D. Saltiel, S. Ungari, and R. L. Abhishek Mukhopadhyay, Jamal Atif. Knowledge discovery with deep rl for selecting financial hedges. In *AAAI: KDF*. AAAI Press, 2021e.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

F. Black. Studies of stock price volatility changes. *Proceedings of the 1976 Meetings of the American Statistical Association, Business and Economical Statistics Section*, 1976.

F. Black and R. Litterman. *Global portfolio optimization*. Financial Analysts, 1992.

A. Blum and M. Furst. Fast Planning Through Planning Graph Analysis. In *IJCAI*, pages 1636–1642, 1995.

A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, Dec. 1997. ISSN 0004-3702.

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323, 2005.

I. Brown and C. Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3):3446–3453, 2012. ISSN 0957-4174.

B. Chaboud, Alain p.and Chiquoine, E. Hjalmarsson, and C. Vega. Rise of the machines: Algorithmic trading in the foreign exchange market. *The Journal of Finance*, 69(5):2045–2084, 2015.

S. Chakraborty. Capturing financial markets to apply deep reinforcement learning, 2019.

C. Chan. *Algorithmic Trading: Winning Strategies and Their Rationale*. Wiley Finance, 2013.

S. Chatzis, A. P. V. Siakoulis, E. Stavroulakis, and N. Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications*, 112:353–371, 2018.

Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In D. Precup and Y. W. Teh, editors, *PMLR*, volume 70 of *Proceedings of Machine Learning Research*, pages 703–711, International Convention Centre, Sydney, Australia, 06-11 Aug 2017. PMLR.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

J. Choo and S. Liu. Visual analytics for explainable deep learning. *CoRR*, abs/1804.02527, 2018.

V. K. Chopra and W. T. Ziemba. The effect of errors in means, variances, and covariances on optimal portfolio choice. *Journal of Portfolio Management*, 19(2):6–11, 1993.

Y. Choueifaty and Y. Coignard. Toward maximum diversification. *Journal of Portfolio Management*, 35(1):40–51, 2008.

Y. Choueifaty, T. Froidure, and J. Reynier. Properties of the most diversified portfolio. *Journal of Investment Strategies*, 2(2):49–70, 2012.

P. Christoffersen, V. Errunza, K. Jacobs, and X. Jin. Is the potential for international diversification disappearing? Working Paper, 2010.

I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization, 2018.

P. Cogneau and G. Hübner. The 101 ways to measure portfolio performance. *SSRN Electronic Journal*, 01 2009.

T. Degris, P. M. Pilarski, and R. S. Sutton. Model-free reinforcement learning with continuous action in practice. *IEEE In ACC*, 2012:2177–2182, 2012.

M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search, 2011a.

M. Deisenroth, D. Fox, and C. Rasmussen. Gaussian processes for data-efficient learning in robotics and control, 2014.

M. P. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *In Proceedings of the International Conference on Machine Learning*, 2011b.

M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning., 2011.

Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28:1–12, 02 2016. doi: 10.1109/TNNLS.2016.2522401.

S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks, 2016.

J. Dias, J. Vermunt, and S. Ramos. Clustering financial time series: New insights from an extended hidden markov model. *European Journal of Operational Research*, 243:852–864, 06 2015.

M. Dixon, I. Halperin, and P. Bilokon. *Machine Learning in Finance: From Theory to Practice*. Springer International Publishing, 2020. ISBN 9783030410674.

A. Dreyer and S. Hubrich. Tail risk mitigation with managed volatility strategies, 11 2017.

F. Ebert, C. Finn, S. Dasari, A. Xie, A. X. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control, 2018.

V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018.

R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189, 1971.

T. G. Fischer. Reinforcement learning in financial markets - a survey. *Discussion Papers in Economics 12*, 2018.

F. Freitas, A. De Souza, and A. Almeida. Prediction-based portfolio optimization model using neural networks. *Neurocomputing*, 72:2155–2170, 06 2009.

S. P. Y. Fung. Optimal online two-way trading with bounded number of transactions. *CoRR*, 2017.

Y. Gal, R. McAllister, and C. E. Rasmussen. Improving PILCO with Bayesian neural network dynamics models, 2016.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, New York, 2nd ed. edition, 2004.

M. Ghalwash, X. H. Cao, I. Stojkovic, and Z. Obradovic. Structured feature selection using coordinate descent optimization. *BMC Bioinformatics*, 17, 3 2016.

S. Ghosal, D. Blystone, A. K. Singh, B. Ganapathysubramanian, A. Singh, and S. Sarkar. An explainable deep machine vision framework for plant stress phenotyping. *Proceedings of the National Academy of Sciences*, 115(18):4613–4618, 2018.

D. B. A. Giuseppe Di Graziano. Optimal trading stops and algorithmic trading. *SSRN*, 2014.

T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber. Exponential natural evolution strategies. *In: Proceedings of Genetic and Evolutionary Computation Conference, pp*, 2010 (2010):393–400, 2010.

L. R. Glosten, R. Jagannathan, and D. E. Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *Journal of Finance*, 48(5):1779–1801, 1993.

M. Goldstein, T. Viljoen, P. J. Westerholm, and H. Zheng. Algorithmic trading, liquidity, and price discovery: An intraday analysis of the spi 200 futures. *The Financial Review*, 49(2):245–270, 2014.

J. W. Goodell, S. Kumar, W. M. Lim, and D. Pattnaik. Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. *Journal of Behavioral and Experimental Finance*, 32(C), 2021.

S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, 05 2017.

D. Guo, P. P. Boyle, C. Weng, and T. S. Wirjanto. Eigen portfolio selection: A robust approach to sharpe ratio maximization. *SSRN*, 2018.

A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-Reinforcement Learning of Structured Exploration Strategies. *arXiv e-prints*, Feb. 2018.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003. ISSN 1532-4435.

I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 545–552. MIT Press, 2005.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. *arXiv e-prints and ICRL 2020*, Dec. 2019.

N. Hansen. The CMA evolution strategy: A tutorial, preprint, 2016.

N. Hansen and A. Auger. CMA-ES: evolution strategies and covariance matrix adaptation. *GECCO 2011*, 2011(1):991–1010, 2011.

N. Hansen and A. Auger. Evolution strategies and CMA-ES (covariance matrix adaptation). *GECCO Vancouver*, 2014(14):513–534, 2014.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

D. Harmon, B. Stacey, Y. Bar-Yam, and Y. Bar-Yam. Networks of Economic Market Interdependence and Systemic Risk. *arXiv e-prints*, Nov. 2010.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009a.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009b.

R. Haugen and N. Baker. The efficient market inefficiency of capitalization-weighted stock portfolios. *Journal of Portfolio Management*, 17:35–40, 1991. doi: http://dx.doi.org/10.3905/jpm.1991.409335.

J. B. Heaton, N. G. Polson, and J. H. Witte. Deep portfolio theory, 2016.

J. B. Heaton, N. G. Polson, and J. H. Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

V. Heidrich-Meisner and C. Igel. Neuroevolution strategies for episodic reinforcement learning. *J. Algorithms*, 64(4):152–168, 2009.

T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

A. Hocquard, S. Ng, and N. Papageorgiou. A constant-volatility framework for managing tail risk. *The Journal of Portfolio Management*, 39:28–40, 2013.

C. Y. Huang. Financial trading as a game: A deep reinforcement learning approach, 2018.

J. Huang, J. Chai, and S. Cho. Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, 14(1):1–24, December 2020.

C. Igel. Evolutionary kernel learning. In *Encyclopedia of Machine Learning and Data Mining*, pages 465–469. Springer, New-York, 2010.

C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.*, 15(1):1–28, Mar. 2007. ISSN 1063-6560.

C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. *Journal of Machine Learning Research*, 9:993–996, 2009.

M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016.

M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization, 2019.

Z. Jiang and J. Liang. Cryptocurrency Portfolio Management with Deep Reinforcement Learning, Dec. 2016.

M. I. Jordan. Lecture notes: Justification for bayes, 2010.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowsi, S. Levine, R. Sepassi, G. Tucker, and H. Michalewski. Model-based reinforcement learning for Atari, 2019.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski. Model-Based Reinforcement Learning for Atari, Mar. 2019.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154. Curran Associates, Inc., 2017.

D. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun. The flash crash: High-frequency trading in an electronic market. *Journal of Finance*, 72(3):967–998, 2017.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 284–292, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 1-55860-419-7.

P. N. Kolm and G. Ritter. Modern perspective on reinforcement learning in finance. *SSRN*, 2019.

V. R. Konda and J. N. Tsitsiklis. On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4): 1143–1166, Apr. 2003. ISSN 0363-0129.

C. Krauss, X. A. Do, and N. Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, 259(2): 689–702, 2017.

M. Kritzman. Six practical comments about asset allocation. *Practical Applications*, 1(3):6–11, 2014.

V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation, 2016.

M. Labadie and C.-A. Lehalle. Optimal algorithmic trading and market microstructure. Working papers, HAL, 2010.

K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin. Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning. *arXiv e-prints*, May 2020.

S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q.

Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1071–1079. Curran Associates, Inc., 2014.

S. Levine and V. Koltun. Guided policy search, 17-19 Jun 2013.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17, 04 2015.

S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 17(1), 03 2016.

X. Li, Y. Li, Y. Zhan, and X.-Y. Liu. Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation, 2019.

Y. Li and W. Ma. Applications of artificial neural networks in financial economics: A survey. In *2010 International Symposium on Computational Intelligence and Design*, volume 1, pages 211–214, 2010.

Liang et al. Adversarial deep reinforcement learning in portfolio management, 2018.

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2016.

S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *CoRR*, abs/1702.01226, 2017.

Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu. Adaptive quantitative trading: an imitative deep reinforcement learning approach, 2020.

I. Loshchilov and F. Hutter. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. *arXiv e-prints*, 1604(Apr):arXiv:1604.07269, Apr. 2016.

S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions, 2017.

M. Madhikermi, A. Malhi, and K. Främling. Explainable artificial intelligence based heat recycler fault detection in air handling unit. In *EXTRAAMAS@AAMAS*, 2019.

S. Maillard, T. Roncalli, and J. Teïletche. The properties of equally weighted risk contribution portfolios. *The Journal of Portfolio Management*, 36(4):60–70, Summer 2010.

A. Malhi, T. Kampik, H. S. Pannu, M. Madhikermi, and K. Främling. Explaining machine learning-based classifications of in-vivo gastral images. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, page 7, 2019-12.

A. Malhi, S. Knapic, and K. Framling. Explainable agents for less bias in human-agent decision making. In *EXTRAAMAS@AAMAS*. Springer, 2020.

A. Mangal and E. A. Holm. A comparative study of feature selection methods for stress hotspot classification in materials. *ArXiv e-prints*, Apr. 2018.

L. Marceau, L. Qiu, N. Vandewiele, and E. Charton. A comparison of deep learning performances with others machine learning algorithms on credit scoring unbalanced data. *CoRR*, abs/1907.12363, 2019.

J.-M. Marin and C. P. Robert. *Bayesian Core: A Practical Approach to Computational Bayesian Statistics (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 387389792.

H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, Mar. 1952a.

H. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952b.

F. Mazrouei and H. Nobanee. Machine learning in finance: A mini-review. *SSRN Electronic Journal*, 02 2020. doi: 10.2139/ssrn.3539038.

R. Mehra and E. Prescott. The equity premium: A puzzle. *Journal of Monetary Economics*, 15 (2):145–161, 1985.

T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, and A. Waibel. Machine learning. *Annual review of computer science*, 4(1):417–433, 1990.

P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):301–312, Mar. 2002. ISSN 0162-8828.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48, pages 1928–1937, New York, New York, USA, 20-22 Jun 2016. PMLR.

T. M. Moerland, J. Broekens, and C. M. Jonker. Model-based reinforcement learning: A survey, 2020.

H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6):619–632, 1991. ISSN 0167-8191.

O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *NIPS*, pages 2775–2785. Curran Associates, Inc., 2017.

A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning. *arXiv e-prints and ICLR 2019*, Mar. 2018.

A. Nan, A. Perumal, and O. R. Zaiane. Sentiment and knowledge based algorithmic trading with deep reinforcement learning, 2020.

S. Niaki and S. Hoseinzade. Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9, 02 2013.

B. Ning, F. H. T. Lin, and S. Jaimungal. Double deep q-learning for optimal execution, 2018.

D. Noureldin and N. Shephard. Multivariate high-frequency-based volatility (heavy) models, 2012.

I. K. Nti, A. F. Adekoya, and B. A. Weyori. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, pages 1–51, 2019.

B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. PGQ: combining policy gradient and q-learning. *CoRR*, abs/1611.01626, 2016.

J.-J. Ohana, S. Ohana, E. Benhamou, D. Saltiel, and B. Guez. Explainable ai (xai) models applied to the multi agents environment of financial markets. In *AAAMAS: EXTRAAMAS*. AAAI Press, 2021.

Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *J. Mach. Learn. Res.*, 18(1):564–628, Jan. 2017. ISSN 1532-4435.

R. Perchet, R. Leote de Carvalho, T. Heckel, and P. Moulin. Predicting the success of volatility targeting strategies: Application to equities and other asset classes, 01 2016.

J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, Mar. 2008.

V. Pong, S. Gu, M. Dalal, and S. Levine. Temporal difference models: Model-free deep RL for model-based control, 2018.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 6638–6648. Curran Associates, Inc., 2018.

L. A. Rastrigin. *Systems of extremal control*. Mir, Moscow, 1974.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1135–1144. Association for Computing Machinery, 2016.

C. P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer Texts in Statistics. Springer, New York, 2007.

T. Roncalli and G. Weisang. Risk parity portfolios with risk factors. *Quantitative Finance*, 16(3): 377–388, Mar. 2016.

A. Rosenfeld and A. Richardson. Explainability in human-agent systems. *CoRR*, abs/1904.08123, 2019.

S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *In: AISTATS, JMLR.org, JMLR Proceedings*, 15:627–635, 2011.

S. A. Ross. The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3): 341–360, 1976.

F. Rundo, F. Trenta, A. L. di Stallo, and S. Battiato. Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24):5574, 2019.

D. Saltiel and E. Benhamou. Feature selection with optimal coordinate ascent (OCA). *arXiv e-prints*, art. arXiv:1811.12064, Nov. 2018.

D. Saltiel and E. Benhamou. Trade Selection with Supervised Learning and OCA. In *Risk forum*, 2019.

D. Saltiel, E. Benhamou, J. J. Ohana, R. Laraki, and J. Atif. Drlps: Deep reinforcement learning for portfolio selection. *ECML PKDD Demo track*, 2020.

A. Samitas, E. Kampouris, and D. Kenourgios. Forecasting stock market crisis events using deep and statistical machine learning techniques. *International Review of Financial Analysis*, 71: 101507, 2020.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 44:206–227, 1959.

S. Schaal. Learning from demonstration. In *NIPS, MIT Press*, pages 1040–1046. NIPS, MIT Press, 1996.

M. Schervish. *Theory of Statistics*. Springer Series in Statistics. Springer, New York, 1996. ISBN 9780387945460.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *arXiv e-prints*, Nov. 2019.

J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel. Trust region policy optimization. In *ICML*, 02 2015a.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *ICLR*, 06 2015b.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1704.06440, 07 2017.

O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005-2019. *arXiv preprint arXiv:1911.13288*, 2019.

D. Shah, H. Isah, and F. Zulkernine. Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies*, 7(2):26, 2019.

L. S. Shapley. A Value for n-Person Games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, number 1 in Proceedings of Machine Learning Research, pages 387–395, Bejing, China, 22-24 Jun 2014. PMLR.

D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 01 2016. doi: 10.1038/nature16961.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, Oct. 2017.

D. Sornette and A. Johansen. Significance of log-periodic precursors to financial crashes. *Quantitative Finance*, 1:452–471, 2001.

W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In *ICML*, volume 70, pages 3309–3318. PMLR, 06-11 Aug 2017.

R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, The MIT Press, 2018.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

T. Théate and D. Ernst. Application of deep reinforcement learning in stock trading strategies and stock forecasting, 2020.

E. Tuv, A. Borisov, G. Runger, and K. Torkkola. Feature selection with ensembles, artificial variables, and redundancy elimination. *J. Mach. Learn. Res.*, 10:1341–1366, Dec. 2009. ISSN 1532-4435.

H. van Hasselt, M. Hessel, and J. Aslanides. When to use parametric models in reinforcement learning?, 2019.

K. Varelas, A. Auger, D. Brockhoff, N. Hansen, O. A. ElHara, Y. Semet, R. Kassab, and F. Barbaresco. A comparative study of large-scale variants of CMA-ES. *PPSN XV - 15th International Conference, Coimbra, Portugal*, 15(2018):3–15, 2018.

D. Vezeris, T. Kyrgos, C. T. P. Schinas, and S. Loss. Trading strategies comparison in combination with an macd trading system. *J. Risk Financial Manag*, 11:56, 2018.

O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019.

H. Wang and X. Y. Zhou. Continuous-time mean-variance portfolio selection: A reinforcement learning framework, 2019.

S. Wang, D. Jia, and X. Weng. Deep reinforcement learning for autonomous driving. *ArXiv*, abs/1811.11329, 2018.

C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Wikipedia. Cma-es, 2018.

R. J. Williams. *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, volume 8. Springer, 1992.

G. Wu. The Determinants of Asymmetric Volatility. *Review of Financial Studies*, 14(3):837–859, 2001.

X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538:142–158, 2020.

F. Z. Xing, E. Cambria, and R. E. Welsch. Natural language based financial forecasting: a survey. *Artificial Intelligence Review*, 50(1):49–73, 2018.

Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid. Practical deep reinforcement learning approach for stock trading, 2019.

Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *AAAI*, pages 1112–1119, New York, 2020. AAAI.

P. Yu, J. S. Lee, I. Kulyatin, Z. Shi, and S. Dasgupta. Model-based deep reinforcement learning for financial portfolio optimization, 01 2019.

A. Zarshenas and K. Suzuki. Binary coordinate ascent: An efficient optimization technique for feature subset selection for machine learning. *Knowledge-Based Systems*, 110:191–201, 2016. ISSN 0950-7051.

M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. *arXiv e-prints and ICML 2019*, art. arXiv:1808.09105, Aug. 2018.

Z. Zhang, S. Zohren, and S. Roberts. Deep reinforcement learning for trading, 2019.

Zhengyao et al. Reinforcement learning framework for the financial portfolio management problem, 2017.

Model based versus model-free reinforcement learning in asset management

Abstract: The promise of machine learning is to learn rules from raw data without any predefined programming rules. Thus, the machine learns and develops a form of intelligence by identifying these rules by itself, purely relying on data. The limits of this new paradigm are that the computer may loop forever because of an infinite number of rules and that found rules may not be stable over time. This is in particular very relevant in quantitative asset management that aims at finding rules and patterns in financial markets, well known to change behavior over time. In this thesis, we examine the central question of whether machine learning should apply with or without models in the field of quantitative asset management. Rather than supporting one or the other thesis, we examine the two approaches in turn. We initially show that machine learning provides some guidance in selecting model decisions to increase overall performance. We then show that machine learning is able to learn rules directly from data using deep reinforcement learning. We prove that this approach generalizes traditional portfolio optimization methods as it lifts the limits of convex optimization and allows for more informed decisions, by directly linking actions to data and extending the agent's states beyond mean and variance. We examined similarities between supervised and reinforcement learning (RL) and demonstrated that the gradient policy method in RL can be presented as a supervised learning method where the labels are the rewards and the loss function is the cross-entropy function. We conclude the thesis with a Bayesian analysis of the CMAES algorithm and the use of Shapley's value to better understand the machine learning model decision process.

keywords: Machine learning in finance, Deep Reinforcement Learning, Quantitative method

Résumé: La promesse de l'apprentissage automatique est d'apprendre des règles à partir de données brutes sans aucune règle prédéfinie. Ainsi, la machine apprend et développe une forme d'intelligence en identifiant elle-même ces règles, s'appuyant uniquement sur des données. Les limites de ce nouveau paradigme sont que l'ordinateur peut boucler indéfiniment étant donné un nombre infini de règles et que les règles trouvées ne continuent pas dans le temps. Ceci est particulièrement important en gestion d'actifs quantitative qui vise à trouver des règles et des modèles sur les marchés financiers connus pour changer de comportement au fil du temps. Dans cette thèse, nous nous posons la question si l'apprentissage automatique doit s'appliquer avec ou sans modèles en gestion quantitative d'actifs. Plutôt que de soutenir l'une ou l'autre thèse, nous examinons tour à tour les deux approches. Nous montrons dans un premier temps que l'apprentissage automatique permet d'améliorer l'efficacité de modèles en sélectionnant les décisions à retenir. Nous montrons ensuite que l'apprentissage automatique est aussi capable d'apprendre des règles directement à partir des données par apprentissage par renforcement profond. Nous prouvons que cette approche généralise les méthodes traditionnelles d'optimisation de portefeuille, supprimant les limites de l'optimisation convexe et permettant des décisions plus sophistiquées au-delà du cadre moyenne et variance. Nous étudions les similitudes entre apprentissage supervisé et apprentissage par renforcement (RL) et montrons que la méthode de stratégie de gradient en RL s'analyse comme une méthode d'apprentissage supervisé avec des étiquettes données par les récompenses et une fonction de perte spécifiée par l'entropie croisée. Nous concluons la thèse par une analyse Bayésienne de la méthode CMAES et l'utilisation des valeurs de Shapley pour mieux comprendre le processus de décision du modèle d'apprentissage automatique.

Mots clefs: Apprentissage machine en finance, Apprentissage par renforcement profond, méthode quantitative