LAGADEC Dorian

ENSAE 3e année
Stage de fin d'études
Année scolaire 2018-2019

Le *reinforcement learning* appliqué à la gestion de portefeuille

Confidentiel

AVISIA,
en mission chez
AEQUAM Capital

Maître de stage :
Matthieu POURBAIX

Paris

17/06/2018 – 31/09/2018

Table des matières

1 Introduction

1.1 Avisia : le cabinet *data centric*

Avisia est un cabinet de conseil spécialisé dans les projets Data. C'est un cabinet de conseil à part dans la mesure où tous les consultants ont une compétence technique, qu'elle soit dans le digital (tracking de parcours client, mise en place de plans de récupération et traitement des données en ligne, ...), dans le marketing quantitatif, dans la data science ou dans les métiers-cibles (banques, assurances, ...). Avisia s'est spécialisé, depuis plus de dix ans maintenant, dans des missions de data complètes, où l'accompagnement va du cadrage de la mission au déploiement des livrables en production, clef en main, en passant par la réalisation technique et managériale.

J'ai rejoint Avisia dans le cadre d'un contrat de professionnalisation en octobre 2018, en tant que consultant Data Scientist Junior au sein de l'offre Risque (Banque/Assurance) gérée par mon manager (et maître de stage) Matthieu Pourbaix. Au cours de cette année d'alternance, j'ai pu, d'un côté, mettre en pratique des savoirs théoriques acquis au cours de mon M2 Data Science à l'ENSAE et, de l'autre, nourrir le besoin d'Avisia d'être au fait des dernières avancées de la recherche académique dans le domaine de la Data Science (réalisation d'articles de vulgarisation et d'ateliers de formation interne). Cet aller-retour entre monde académique et monde professionnel a été extrêmement fécond pour ma rélexion sur mon futur et sur mon domaine de compétence.

Au cours de ma période d'alternance, j'ai eu l'occasion de travailler sur différents sujets. Dans un premier temps, ma prise en main des outils de l'entreprise (Dataiku surtout) m'a laissé en support sur des micro-missions au sein des missions réalisées par mon manager ; ensuite, j'ai mis à profit ma connaissance basique de la théorie des graphes pour présenter un projet à la DGCCRF (Direction Générale de Concurrence, de la Consommation et de la Répression des Fraudes) sur le traitement des faux avis en ligne (source de concurrence déloyale) ; surtout, j'ai longuement travaillé sur un projet de scoring de crédit pour le compte de la Banque Casino, qui propose avec des partenaires comme CDiscount ou Lydia des services de crédit à la consommation ; enfin, j'ai travaillé ponctuellement sur des ateliers de formation interne (introduction au Natural Language Processing - NLP - , introduction à la Blockchain, ...) .

Ma période à plein temps chez Avisia, à partir de la fin des cours à l'ENSAE (mi-juin 2019), a été dédiée à une mission de recherche pour le compte d'un nouveau client d'Avisia : la société de gestion Aequam Capital.

1.2 Mission longue : Aequam Capital

Aequam Capital est une société de gestion quantitative employant une dizaine de personnes. Située en face du Palais Brongniart, elle a été créée par des professionnels de la finance qui en ont fait un spécialiste de la gestion multi-facteurs. Aequam Capital a lancé il y a plus d'un an un fonds d'investissements appelé Aequam Dynamic Premia Equity (ADPE), un fonds *long-only* (pas de vente à découvert) n'impliquant que des actions européennes. Le fonds est proposé à des clients institutionnels ainsi qu'à des particuliers par l'intermédiaire de Conseils en Gestion de Patrimoine (CGP). De nature basse fréquence, le portefeuille est modifié tous les mois et les ordres sont dictés par un algorithme propriétaire breveté dont l'objectif est de maximiser la performance sous des contraintes de volatilité et de contrôle des pertes.

L'algorithme en question est donc primordial dans la génération de performance, donc dans le succès de la société. Par conséquent, il est constamment mis à l'épreuve et la recherche est permanente pour s'assurer de la stabilité, voire de la croissance, des performances passées. Dans ce cadre, le Chief Investments Officer (CIO) Thierry Béchu et le Portfolio Manager (PM) Pierre Colonna travaillent avec un chercheur du Massachusetts Institute of Technology (MIT), Florian Berg. C'est ce trio qui m'a accompagné et m'a suivi de près pendant cette mission de recherche.

1.3 Le projet : le *reinforcement learning* appliqué à la gestion de portefeuille

"Les performances passées ne préjugent pas des performances futures", voilà la phrase que tous les gérants de portefeuille doivent accoler à leur publication de performance, en accord avec la régulation de l'AMF (Autorité des Marchés Financiers). Autrement dit, il est très simple d'observer le comportement d'une stratégie de trading sur les jours passés mais en aucun cas on n'est assuré de la réplication dudit comportement sur les périodes futures. Les prix des actions sont par essence très difficiles à prédire et il serait bien prétentieux de croire qu'un algorithme bien calibré sur le passé donnerait des résultats parfaits sur le futur.

Or, on a beau parler d'intelligence artificielle, de deep learning, de machine learning, ... dans les médias - même spécialisés -, la réalité est que nombre de modèles statistiques, fussent-ils avancés, sont statiques. Le réseau de neurones convolutionnels le plus complexe n'est jamais qu'un prédicteur, entraîné sur un jeu de données d'apprentissage, validé sur un jeu de données de test, fixé dans le temps ; par essence, il est très capable d'exploi-

ter ce qu'il a appris, mais réagit très mal à ce qu'il n'a jamais rencontré. Appliqués à des séries chronologiques non stationnaires, non saisonnières, difficilement modélisables comme les données financières (nul n'est besoin ici de rappeler la non normalité des rendements financiers), cela signifie que le modèle a toujours un temps de retard puisqu'il lui faut mal réagir, se tromper, puis réapprendre de son erreur pour, supposément, mieux réagir la fois suivante. Ce retard structurel induit une perte de performance qui peut devenir très problématique (très coûteuse) dans le cadre du trading.

Dans le cadre de la recherche d'Aequam Capital, mes encadrants ont donc souhaité explorer des algorithmes réellement dynamiques, qui ne se contentent pas d'exploiter un apprentissage passé mais contiennent une dimension non négligeable d'exploration, d'"apprentissage sur le tas". Toute une branche du machine learning, que l'on appelle Online learning (apprentissage en ligne), aborde cette question de l'apprentissage continu ; aiguillés par leur chercheur Florian Berg, la voie qui a été choisie est celle de l'apprentissage par renforcement, ou "Reinforcement Learning". En quelques mots (le modèle et ses spécifications seront abordées plus précisément plus bas), il s'agit d'un système dans lequel un agent observe un environnement, en fonction duquel il prend une décision et choisit une action à opérer, pour laquelle il obtient, à l'étape d'après, une récompense ; cette récompense lui sert à évaluer l'action qu'il a choisie et à adapter, au coup suivant, sa décision aux récompenses passées. En fonction de l'environnement observé, de sa proximité aux environnements précédemment observés, l'agent prend plus ou moins fréquemment des décisions aléatoires ou improbables, qui lui permettent de ne pas s'enfermer dans l'exploitation du connu et, au contraire, d'explorer de nouveaux champs de possibilité. Comme souvent, cette configuration théorique s'inspire de l'apprentissage humain - on peut penser à un élève qui reçoit une note sur un travail qu'il a effectué, ou une souris qui reçoit un bout de fromage si sa mémoire ne lui a pas fait défaut, ou encore à un joueur qui reçoit des points en fonction de la manière dont il a résolu un niveau de jeu vidéo. Au passage, c'est sur ce dernier exemple, particulièrement adapté, que l'apprentissage par renforcement a acquis sa notoriété.

Appliqué à la problématique d'Aequam Capital, la question fut la suivante : est-il possible de créer un algorithme - l'agent - qui observe des prix d'actions et des indicateurs macroéconomiques - l'environnement -, décide de l'allocation du portefeuille et achète ou vende des actions pour obtenir cette allocation - l'action - puis observer les rendements corrigés de la volatilité - la récompense - ? Et si oui, un tel algorithme génère-t-il de la performance ? Cette question, vaste, fut découpée en sous-questions qui occupèrent la totalité de ma mission chez Aequam Capital.

2 Mise en place du projet : considérations pratiques, éléments théoriques et proposition d'une démarche de recherche

2.1 Aequam Capital : mandat de gestion et méthode

Avant tout, il est important de placer le contexte du projet car ce dernier impose des contraintes - de gestion, de marketing, de communication - qui ne doivent pas être ignorées sous peine de frapper de nullité tout modèle irréalisable en pratique.

Aequam Capital est une société de gestion qui propose un fonds, Aequam Dynamic Premia Equity (ADPE) dont le mandat est le suivant : gérer dynamiquement le portefeuille de sorte à répliquer la performance de long terme du marché, tout en réduisant la volatility et la perte maximale ; en d'autres termes, il s'agit d'essayer de faire comme le marché (indice de référence Eurostoxx 50) lorsque ce dernier est haussier, et de réduire la baisse lorsque ce dernier est baissier.

L'univers d'investissement est l'Eurostoxx 600 (actions des 600 plus grandes capitalisations boursières en Europe) et les contraintes sont assez fortes : *long-only* (pas d'achat à découvert, donc pas de problématique de REPO - Sales and Repurchase Agreement), actions européennes, pas de produits dérivés (en particulier pas d'options, donc des produits linéaires), un univers particulièrement liquide donc des hypothèses de coûts de transaction assez stables (de l'ordre de 0.1%).

En parallèle, il existe une forte contrainte de continuité. Les fonds d'investissements étant strictement réglementés, des documentations sont émises et validées par l'AMF et il est important que le produit financier existant soit similaire à celui qui a été vendu. Par conséquent, il faut partir du modèle actuel si l'on veut en faire un second : le modèle actuel est dit "quantamental" en ce qu'il combine analyse quantitative et analyse fondamentale. Très simplement, l'algorithme propriétaire prend en entrées des données financières et des données macroéconomiques, les retraite (selon une formule propriétaire), les combine et renvoie un nombre entre 0 et 1 (arrondi au dixième près) qu'Aequam Capital appelle le "Quantamental Ratio"). Ce quantamental ratio est un indicateur de risque ou de confiance envers le marché : à 0, l'algorithme est très pessimiste quant au marché et suggère de garder du cash (la vente à découvert n'étant pas autorisée par le mandat de gestion) ; à 1, l'algorithme est très optimiste quant au marché et suggère d'être complètement investi (l'effet de levier n'étant pas permis puisqu'aucun produit dérivé n'entre dans la composition du portefeuille). Ce quantamental ratio est publié par Aequam Capital tous les mois

et commande - modulo les décisions des gérants - la réallocation à effectuer, opération qui a lieu en début de mois. Cette contrainte temporelle est très forte car elle implique de rester stoïque en cas de crise ou de marché très haussier ; sans exagérer, elle induit à tout le moins que les gérants ratent nécessairement des bons points d'entrée ou de sortie sur le marché. Il est à noter que cette contrainte a été choisie telle quelle mais qu'elle a dès le départ été sujette à modification ; en revanche, pour satisfaire la problématique de continuité (étant difficile d'expliquer à des clients que pendant un an le portefeuille était changé mois par mois, puis du jour au lendemain de façon erratique tous les jours, deux jours, cinq jours etc), il était clair dès le départ du projet que l'algorithme final ne serait pas autorisé à échanger des actions à n'importe quelle fréquence. L'ambition du fonds étant sur du moyen à long terme, la fréquence des échanges ne devrait pas dépasser 5 par mois.

Enfin, il faut maintenant présenter les produits échangés par Aequam Capital : Aequam Capital s'est spécialisé dans la gestion multi-facteurs et dans l'investissement dans les primes de risque. De la même manière qu'il existe des ETF (Exchange-Traded Fund), des *trackers* permettant de répliquer la performance d'un indice (CAC 40, S&P 500, Nikkei, Eurostoxx 50, ...), il existe ce qu'on appelle des facteurs de risque, ou *risk premia*. Un facteur de risque est un panier d'actions dont les actions possèdent toutes des propriétés similaires et qui donc exposent l'investisseur à un facteur de risque donné. Par exemple, un facteur "Value" contient des actions dépréciées (relativement à un indicateur, Price to Book par exemple, ou Price to Earnings) qui ont tendance à surperformer le marché en période haussière ; ou encore, un facteur "Momentum" contient des actions qui ont très bien performé dans la dernière période et dont on considère que la performance sera persistante sur la prochaine période. Il en existe de nombreux mais seuls 6 sont utilisés par Aequam Capital, pour des raisons de précision de définition et de performance : Value, Profitability, Low Vol, Quality, Carry et Momentum. Leur définition se trouve dans le tableau ci-dessous, extrait d'un document commercial d'Aequam Capital :

FIGURE 1 – Définition des facteurs de risque

Cet univers, désormais restreint, d'investissement, suggère en filigrane une seconde contrainte assez forte : celle de pouvoir justifier l'allocation de l'algorithme. Certaines de ces primes ont un caractère défensif (préservation de capital, faible corrélation à la performance globale du marché), d'autres ont un caractère offensif (surperformance du marché en période haussière, sous-performance du marché en période baissière) et il est difficile d'expliquer à des clients pourquoi on mixerait les deux parfois, on les séparerait d'autre fois, etc. En d'autres termes, l'interprétabilité du modèle, de plus en plus faible à mesure que la complexité du modèle augmente, est cruciale et joue un rôle de validation ou d'invalidation qui peut être décisif.

2.2 Les données disponibles et leur transformation

La question des données est cruciale dans tout problème d'apprentissage statistique. Dans un problème de finance de marché, il y a rarement un problème de pénurie de données ou de données manquantes, ou erronées, ou corrompues. Au contraire, les fournisseurs de données financières comme Bloomberg ou Reuters ont une quantité phénoménale de données à proposer et la problématique est plutôt celle de la sélection. L'interconnexion est une caractéristique forte des marchés financiers, donc le nombre de variables ayant une corrélation avec la variable cible (pouvant expliquer le rendement, donc ayant un pouvoir

prédictif) est immense et il est évidemment impossible de toutes les prendre en compte. Nous avons donc été contraints de faire des choix, basés sur l'expérience des gérants et sur l'expertise du chercheur. Voici les données sélectionnées *a priori* comme utilisables par l'algorithme :

— Les prix des facteurs : les prix ayant une dynamique propre, il est évident - et démontré - que les observer est utile pour faire un choix d'allocation gagnant ;
— Des indicateurs macroéconomiques : indicateurs d'états économiques, de tendances à venir, de changements etc, ils offrent des regards exogènes sur les marchés financiers. Parmi eux :
    1. Des données de volatilité qui peuvent indiquer un environnement stressé ;
    2. Des données de volumes échangés qui renseignent sur les comportements des agents évoluant sur les marchés ;
    3. Des données de surprises économiques qui informent sur les chocs informationnels et leurs impacts sur les prix de marché.

Se pose bien sûr la question de la transformation des données. A ce sujet, il m'a fallu jongler entre la théorie apprise lors de ma formation, les contraintes d'implémentation pratiques et le caractère non-aléatoire, autocorrélé des séries temporelles à ma disposition.

Prenons un exemple simple : les prix de marché, c'est de notoriété publique, sont des séries hautement non stationnaires. Elles font montre de persistence, d'autocorrélation, d'intermittence, etc ; dans le même temps, les transformer en séries stationnaires a-t-il du sens ? Conserver le bruit dans ces séries conduit à l'écueil suivant : si le modèle assimile ce bruit comme de la stationnarité, alors les actions prescrites par le modèle ont tendance à être également bruitées et donc l'allocation qui s'ensuit va être chaotique, donc non implémentable (les coûts de transaction cumulés devenant vite prohibitifs) et surtout non vendable (comment expliquer que l'on change l'allocation du tout au tout d'un jour sur l'autre ?).

Des transformations non stationnaires peuvent donc être envisagées pour assurer une continuité temporelle des décisions prises par l'algorithme et même si un point d'honneur a été mis à considérer des données débruitées et stationnarisées, des choix ont dû être faits pour tempérer ce besoin.

En outre, on observe que ce n'est pas parce qu'une transformation ne fait pas de sens mathématiquement qu'elle n'apporte pas d'information. Si de nombreux traders suivent et

utilisent tous les jours dans leurs stratégies une transformation, alors cette transformation a un impact sur les prix et donc l'utiliser apporte de l'information et un pouvoir prédictif sur les rendements. Cette remarque nous a conduit a calculer des écarts-types sur des prix (non stationnaires) plutôt que sur des rendements (potentiellement plus stationnaires) par exemple.

Voici une liste non exhaustive des transformations de données qui ont été utilisées lors de ce projet. Bien souvent, ces transformations ont été empruntées au domaine de l'analyse technique, domaine dans lequel le CIO Thierry Béchu a une certaine expertise :

— **MACD** (Moving Average Convergence Difference) : difference entre deux moyennes mobiles exponentielles sur un prix (une de courte période, une de longue période). Cet indicateur est très utilisé par les analystes techniques et le signe de cette différence apporte une information sur le prix (un changement de signe indique un retournement de tendance) ;

— **Stochastique** : indicateur de positionnement de prix sur la période récente. On fixe un paramètre de période, n, on calcule le prix maximum sur la période, H_n, et le prix minimum sur la période B_n et le stochastique se calcule selon la formule $Stochastique = 100 \times \frac{Prix - B_n}{H_n - B_n}$. Lorsque le stochastique est proche de 100, le prix est proche de son maximum ; inversement, lorsque le stochastique est proche de 0, le prix est proche de son minimum sur la période.

— **RSI** (Relative Strength Index) : indicateur comportemental qui indique la tendance récente et son potentiel excès. A nouveau, on fixe un paramètre de période n. Sur cette période, on calcule une moyenne mobile exponentielle des hausses $\bar{H}_n$ et une moyenne mobile exponentielle des baisses, mise en valeur absolue, $\bar{B}_n$. Le RSI se calcule comme suit : $RSI = 100 \times \frac{\bar{H}_n}{\bar{H}_n + \bar{B}_n}$. Lorsque le RSI est proche de 100, le titre est très régulièrement en hausse, potentiellement suracheté ; lorsque le RSI est proche de 0 en revanche, le titre est très régulièrement en baisse, potentiellement survendu.

Sans aller plus dans le détail, on voit que chacune de ces transformation contient des paramètres (taille de la fenêtre, durée des moyennes mobiles, ...), on voit également le nombre de données disponibles, et donc on imagine la quantité possible de jeux de données à considérer. Il a donc fallu faire des choix, dès avant le modèle, dans la sélection des données et de leurs transformations.

Une manière de réduire le bruit et de réduire le nombre de variables à considérer a été l'**Analyse en Composantes Principales** (ACP).

L'ACP est une méthode de réduction de dimension : elle vise à réduire l'information donnée par un large nuage de points en grande dimension en trouvant un équilibre entre un faible nombre de dimensions et une forte variance expliquée. L'ACP transforme des variables liées - dans notre cas, les prix d'un facteur sont corrélés avec les prix d'autres facteurs, avec des indicateurs macroéconomiques, ... - en des variables décorrélées appelées composantes principales, ou facteurs. Utilisée dans une optique descriptive, l'ACP permet d'isoler des sources de variances indépendantes pour expliquer, au sens mathématique de la variance, mais également au sens littéral de la compréhension, des mouvements.

L'ACP diagonalise la matrice de variance-covariance d'un nuage de points donné et trouve un vecteur tel que la variance de la projection du nuage sur ce vecteur soit maximisée : ce vecteur est le premier facteur. Puis, il trouve, parmi les vecteurs orthogonaux au premier facteur (selon le produit scalaire canonique) un vecteur tel que la variance de la projection du nuage sur l'espace vectoriel engendré par ces deux premiers vecteurs soit maximisée : c'est le second facteur. Et ainsi de suite. Le gain en variance expliquée décroît, par construction, à chaque nouveau facteur et de nombreux critères existent pour savoir combien de facteurs retenir (moins de facteurs que de variables évidemment, sinon la méthode a peu de sens). C'est une méthode très utilisée en finance, notamment parce qu'elle conserve relativement le pouvoir descriptif tout en réduisant drastiquement le nombre de variables (à titre d'exemple, ces mots sont fortement inspirés de mon précédent rapport de stage, sur l'études des produits dérivés sur dividendes chez Melanion Capital, qui impliquait également l'ACP).

Nous avons donc appliqué l'ACP sur 15 variables (les 6 séries de prix stationnarisées - en rendement - et 9 indicateurs macroéconomiques cités ci-dessus également stationnarisés) standardisées (pour éviter un effet-taille caractéristique de l'ACP). L'analyse de la variance expliquée par les composantes principales, dont les participations sont exhibées en figure 2, conduit à ne garder que 4 composantes principales (d'après le critère du coude). En outre, lorsqu'on observe le tableau des variances expliquées cumulées, on voit que 4 composantes principales suffisent à expliquer près de 75% de la variance du nuage de points, ce qui est amplement suffisant pour notre problématique.

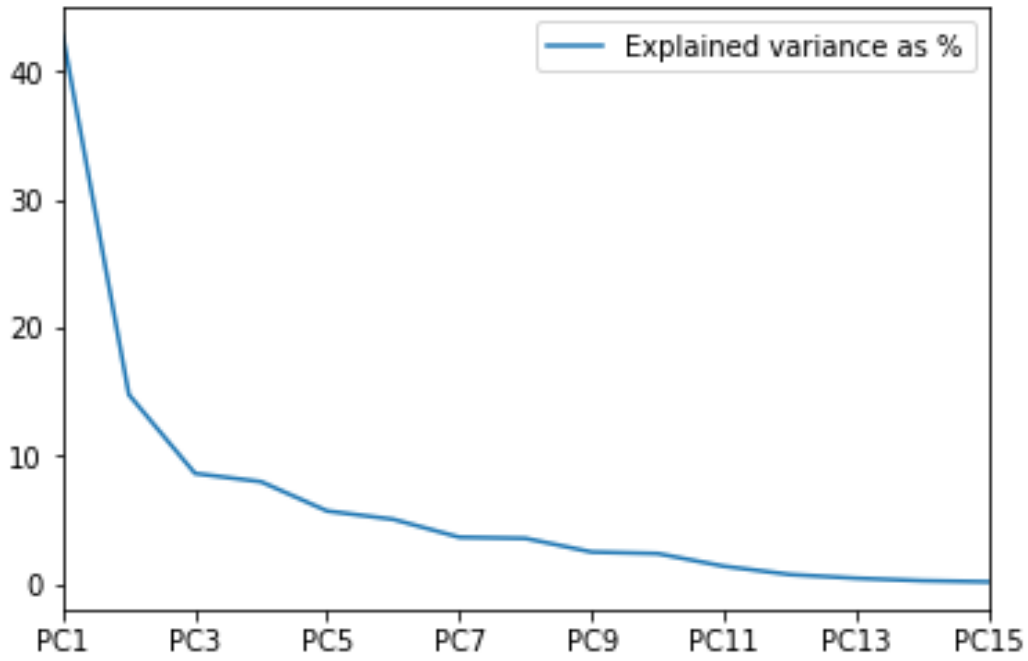


FIGURE 2 – Variance expliquée par les composantes principales

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Explained variance as %	42.858005	14.781154	8.627475	7.981220	5.692024	5.043410	3.641506	3.557714	2.491418
Cumulative explained variance as %	42.858005	57.639160	66.266634	74.247854	79.939878	84.983288	88.624794	92.182509	94.673927

FIGURE 3 – Variance expliquée par les premières composantes principales

Mais cet ensemble de composantes principales n'est pas satisfaisant en l'état. En effet, notre problème est un problème séquentiel impliquant des séries temporelles. Par conséquent, un retraitement de données ne peut pas impliquer des données futures. L'ACP étant une opération matricielle, elle opère un retraitement de données en fonction de tout le nuage de point, donc de tout l'historique, ce qui est gênant. L'option qui a donc été choisie a été de faire une ACP "mobile", c'est à dire faire une ACP à chaque date, ne prenant en compte que les données passées. L'opération est un peu lourde mais permet de vérifier des contraintes de faisabilité, dans une optique de production. Ceci dit, au début de l'historique, on peut imager que le peu de données disponibles donne des composantes principales bien différentes que celles obtenues ci-dessus. Le graphique en figure 4 compare les valeurs des composantes principales de l'ACP statique a celle de l'ACP mobile. Comme on peut s'y attendre, la différence converge vers 0 à mesure que le temps converge

vers le temps présent. D'après le graphique, on voit qu'à partir de 2014 on peut sans peine utiliser l'ACP mobile sans que cela soit problématique.

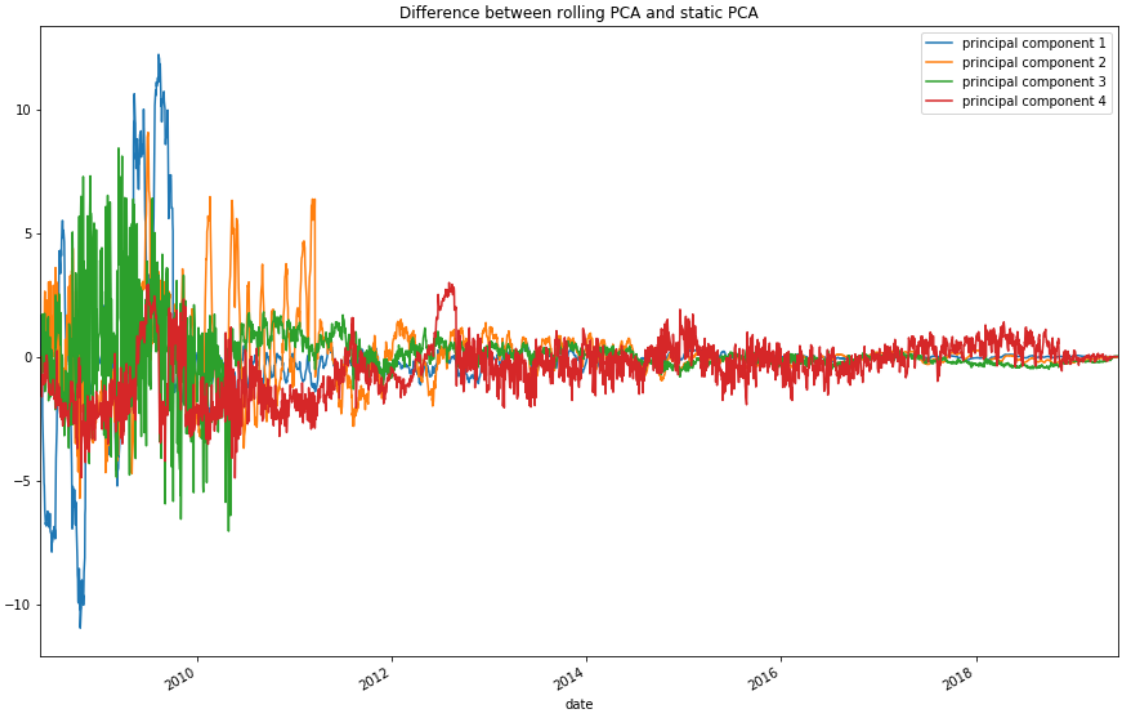


FIGURE 4 – Différences entre composantes principales statiques et mobiles

Ces composantes principales viennent ajouter à la liste des jeux de données disponibles un nouveau jeu de données, de faible dimension, potentiellement débruité. Nous verrons par la suite s'il offre des perspectives intéressantes à notre problème. Il est temps de comprendre ce qu'est le Reinforcement Learning et comment on peut l'appliquer à notre problème.

2.3 Le Reinforcement Learning : une brève introduction

L'apprentissage par renforcement (que nous noterons reinforcement learning) est une branche très récente du machine learning. Le site d'Open AI, organisation connue pour fournir des "environnements" (nous allons voir ce que signifie ce mot) open-source de reinforcement learning, donne à cette adresse des articles importants dans ce domaine : à quelques exceptions près (des publications en 1990, 2000 ou 2008), tous ont été publiés après 2010. Et pour cause, le reinforcement learning a été popularisé par un article intitulé *Playing Atari with Deep Reinforcement Learning* et publié par des ingénieurs de Google DeepMind. Dans cet article, les auteurs montrent comment il est possible d'entraîner un ordinateur à jouer à n'importe quel jeu Atari. L'ordinateur commence par faire des erreurs puis s'améliore graduellement - très vite - jusqu'à devenir imbattable. L'idée est de

faire jouer l'ordinateur un grand nombre de fois et de le récompenser pour ses bonnes actions tout en le punissant pour ses mauvaises actions. Comme un humain, l'ordinateur va fuir les actions qui sont susceptibles de lui amener une punition et préférer des actions susceptibles de lui procurer une grande récompense. Cette idée, simple à vulgariser, extrêmement performante sur des jeux simples, a beaucoup plu et a été developpée par la suite de nombreuses fois. Alpha Go, l'ordinateur célèbre pour avoir battu 4-1 en 2016 Lee Sedol (considéré comme le meilleur joueur de Go du monde de 2000 à 2010) a également été développé par DeepMind avec un apprentissage par renforcement. Bref, la réputation du reinforcement learning n'est plus à faire tant les résultats qui ont été obtenus sont spectaculaires.

L'apprentissage par renforcement ne s'oppose pas à l'apprentissage supervisé ou non supervisé : au contraire, il se surajoute, se superpose. C'est une manière de transformer un prédicteur statique en un prédicteur séquentiel. Comme il a été écrit plus haut, les algorithmes classiques (random forest, réseau de neurones, ...) sont à l'origine statiques et cette staticité pose problème dans un grand nombre de cas, surtout lorsqu'on cherche à travailler sur des comportements essentiellement adaptatifs (les comportements humains typiquement). Disons quelques mots du cadre général dans lequel on s'inscrit lorsqu'on développe une méthode de reinforcement learning. Dans ce cadre théorique, on trouve :

— Un **agent** : c'est celui qui prend les décisions et dont on veut optimiser le processus de décision. Schématiquement, c'est l'humain que le système se propose de remplacer ;
— Un **environnement** : c'est le cadre dans lequel l'agent évolue, qui contraint les possibilités d'action et de perception ;
— Des **états**, que l'agent est capable de percevoir. Ce sont les indicateurs qu'il va rapprocher de ce qu'il "connaît" (ou non), en fonction desquels la meilleure décision va être choisie ;
— Des **actions** par lesquelles l'agent interagit avec son environnement ;
— Une fonction de **récompense** cruciale car c'est elle qui va conduire l'agent à améliorer ses actions par la suite. Mieux la fonction de récompense est construite, plus l'agent va apprendre vite de ses erreurs ;
— Une **politique** qui dicte la façon de sélectionner l'action selon l'état de l'environnement

Le but, dans ce cadre, est de trouver la politique qui va maximiser l'ensemble des récompenses reçues. On voit d'ores et déjà les difficultés inhérentes à tout projet de rein-

forcement learning : définir précisément chacun de ces objets, tordre le problème concret pour le faire entrer dans ce cadre contraignant, définir une fonction de récompense qui valorise justement chaque aspect positif d'une décision tout en pénalisant ses aspects négatifs (par exemple en finance : un trade est-il bon s'il génère un rendement tout en exposant l'investisseur à un risque important ?), ...

Ce cadre théorique a beau être relativement contraignant, il reste très vague. De nombreuses lignes de fracture traversent le reinforcement learning et permettent de séparer différents champs de recherche, différentes méthodes d'optimisation. En voici quelques unes :

— ***Model-based* ou *model-free*** : cette ligne de partage correspond grossièrement à celle qui sépare les statistiques paramétriques des statistiques non paramétriques. Dans la plupart des cas, on peut voir l'environnement comme une chaîne de Markov sur laquelle l'agent se déplace : discrète ou continue, il est tentant d'en apprendre la probabilité de transition. Si l'on choisit des distributions dont on cherche les meilleurs paramètres, on est dans le cadre *model-based* ; si l'on opte pour une méthode *trial-and-error*, on est dans le cadre *model-free* ;

— ***On-policy* ou *off-policy*** : ici, on se demande si l'algorithme apprend de ses actions effectives ou des actions qu'il aurait pu effectuer. Autrement dit, si l'on est capable, à une date t, de voir l'effet de chaque action sur l'environnement et donc sur la récompense, on peut avoir un apprentissage *off-policy* où l'on apprend de tout ; en revanche, si l'on ne peut voir que l'impact des actions ayant vraiment été effectuées, alors l'apprentissage est nécessairement *on-policy*. On peut penser, pour ce dernier cas, à un médecin qui doit choisir entre plusieurs médicaments pour des patients : on ne peut observer que l'effet du médicament effectivement pris.

— ***Value method* ou *policy method*** : cette distinction sépare les algorithmes qui cherchent à trouver la politique qui va maximiser la somme actualisée des récompenses (optique de long terme) - c'est la *policy method* - des algorithmes qui cherchent à toute date l'action qui va maximiser la récompense suivante - *value method*.

Ceci constitue un bref aperçu de l'idée qui sous-tend tout le domaine du reinforcement learning. Pour aller plus loin, de nombreuses ressources sont disponibles en ligne et une section y est dédiée dans la bibliographie (elle est intitulée "Introduction au rein-

forcement learning"). Cette section regroupe les articles qui m'ont permis de rapidement cerner le fonctionnement et les enjeux du reinforcement learning, avant d'entreprendre une implémentation adaptée au problème d'Aequam Capital.

2.4 Intersection entre théorie générale et cas d'usage particulier : la démarche proposée

D'emblée, on voit bien comment le cadre du reinforcement learning s'applique à une problématique d'investissement ou de trading. Les parallèles sont flagrants : dans leur métier, les traders observent constamment les données financières pour prendre des décisions à l'achat ou à la vente et ces décisions sont presque immédiatement récompensées par du profit ou de la perte, les enjoignant à analyser *a posteriori* leurs trades pour voir quel raisonnement a été juste, quel raisonnement a été biaisé et pourquoi, etc. L'application est presque naturelle et nombreux sont ceux qui ont comparé la spéculation à un jeu, champ d'application par excellence du reinforcement learning.

De fait, dans la littérature, nombreux sont les articles qui tentent d'utiliser le reinforcement learning pour trouver des stratégies systématiques de trading. Là encore, une section est dédiée à ces articles dans la bibliographie et est intitulée "Le reinforcement learning appliqué à la finance" ; bien que riche pour si peu d'années, cette section est loin d'être exhaustive et je n'ai cité que les articles ayant participé de ma réflexion lors de ce projet. On y trouve, notamment, des articles qui traitent de l'exécution optimale d'ordres ([11]), de nombreux articles qui traitent de la problématique - simple à faire coïncider avec le schéma général - de *day trading* ([12], [15], [16]), ou encore des articles traitant d'une problématique plus proche de la notre, à savoir de la gestion multi-actifs et de la gestion de portefeuille ([18] et [19]).

Cependant, lorsqu'on se penche de plus près sur notre problème, on se rend compte que des ajustements sont nécessaires. Pour un gérant de portefeuille, on l'a vu plus haut, des contraintes d'explicabilité, de cohérence temporelle également et de faisabilité existent. Par exemple, que doit mesurer notre récompense ? Une performance ? Si oui, doit-elle être en valeur brute ou comparée à une référence (un indice boursier par exemple) ? La volatilité du portefeuille doit-elle être prise en compte ? Autant de questions auxquelles il faut répondre pour créer un modèle si l'on est dans une optique de production. Etant dans un projet qualifié de recherche par la société Aequam Capital, la progression envisagée a été la suivante :

1. D'abord, voir si une stratégie dite "*long-flat*" était possible. C'est-à-dire, avant de chercher des poids de facteurs optimaux à toute date, voir s'il est possible d'avoir une stratégie plus performante que la référence avec un mode "on-off", *i.e.* un portefeuille soit investi complètement en cash ("on"), soit investi complètement dans un portefeuille équi-pondéré de facteurs ("off"). Il s'avère que cette étape, envisagée comme préliminaire, s'est avérée plus complexe et plus chronophage que prévu ;
2. Ensuite, voir si une stratégie légèrement plus libre était plus performante que la première : partir du portefeuille équipondéré et changer un à un les poids des facteurs pour voir si des tendances ou des *politiques* pouvaient être détectées par l'algorithme ;
3. Enfin, aborder l'objectif final, à savoir créer un algorithme de reinforcement learning qui puisse à toute date donner des éventuels changements de portefeuille pour avoir un portefeuille le plus performant possible.

Nous allons voir plus bas en quoi le développement effectif du projet a suivi ce fil rouge tout en s'écartant ponctuellement de l'idée initiale.

3 Exécution du projet : une évolution incrémentale conduisant à une compréhension toujours plus précise du problème

3.1 Algorithme et spécifications

3.1.1 Le point de départ : librairies et codes publics

Comme bien souvent dans un projet qui implique une forte dose de programmation, il convient de s'appuyer sur la quantité de codes disponibles en ligne, notamment sur la plateforme Github. En reinforcement learning, l'idée ayant beau être simple, la programmation concrète des éléments - environnement, agent, action, états, récompense etc - et de leurs interactions est assez complexe (en tout cas trop pour un projet de 14 semaines) mais de nombreuses librairies sont disponibles. Parmi eux, OpenAI Gym fournit les environnements et les codes exemples pour tester les jeux Atari, puis les librairies partiellement rendues publiques par Facebook et Google comme Facebook Horizon, Google Dopamine ou DeepMind TRFL implémentent différents algorithmes connus de reinforcement learning. Enfin, des comptes Github personnels comme celui de Denny Britz (ex Google Brain, Stanford, Berkeley) ou celui de Tal Perry (développeur chez Google, fondateur de la plateforme d'annotation de texte par NLP LightTag.io) m'ont été précieux pour avoir une vue d'ensemble des algorithmes de reinforcement learning et pour m'inspirer d'applications simples. Car l'écueil des librairies très puissantes (comme celles de Google ou de Facebook) pour un projet comme celui-ci est qu'elles sont très complètes, très puissantes en termes de vitesse de calcul, mais aussi très contraignantes en termes d'implémentation. Il est difficile, pour la plupart, de les installer ou de les faire fonctionner avec un ordinateur portable basique ; il peut également être difficile d'explorer l'"intérieur" de l'algorithme pour en comprendre le comportement. En revanche, les parcourir et les étudier m'a été d'une grande aide pour créer mes propres codes. Ce sont ceux-là que je vais désormais décrire.

3.1.2 Version initiale du projet : trois programmes

Chaque programme correspond à une partie de l'infrastructure inhérente à tout algorithme de reinforcement learning.

trading environment Ce programme contient une classe appelée TradingEnvironment et définit, comme son nom l'indique, un environnement de trading, l'agent qui évolue à l'intérieur ainsi que leurs interactions. Cette classe nécessite beaucoup d'arguments pour être instanciée : un tableau de données contenant les prix des actifs ainsi que les

variables susceptibles d'apporter de l'information pour la prédiction de ces prix (les variables citées ci-dessus, transformées), les poids initiaux du portefeuille de facteurs, la métrique de récompense utilisée, l'horizon de temps considérée, l'hypothèse sur les coûts de transaction, ...

Une instance de cette classe peut être initialisée, réinitialisée, on peut calculer à tout moment l'historique des transactions effectuées, des performances, ... En outre, on peut lui faire faire un "pas" qui va changer le portefeuille, changer donc les états observés par l'agent, calculer la fonction de récompense, ...

Il faut ici dire quelques mots de la fonction de récompense. Cette fonction est essentielle dans tout problème de reinforcement learning car c'est elle qui influence l'adaptation de l'agent à son environnement et à ses actions passées. Pour faire le parallèle avec les notions d'apprentissage supervisé ou non supervisé, c'est comme si nous décidions nous même du label, presque "à la louche". Ce sujet est tellement crucial que toute une partie du reinforcement learning, que l'on appelle l'*inverse reinforcement learning* (IRL), est consacrée au renversement du paradigme et à l'apprentissage de la fonction de récompense en fonction d'actions passées supposées optimales. Cette fonction de récompense doit contenir tous les phénomènes que nous voulons influencer. Dans notre cas, notre fonction de récompense doit conduire l'algorithme a effectué peu de changements de portefeuilles, au bon moment - on ne souhaite pas, à l'instar des *day traders* profiter des mouvements stochastiques des prix de marché, mais à l'inverse investir sur des tendances macroéconomiques de long terme -. Dès le début du projet, une discussion a eu lieu entre mes encadrants à ce sujet, qui illustre la tension entre monde académique et monde professionnel. Quand l'un soutenait que la métrique de récompense devait faire intervenir le ratio de Sharpe, défini comme $\frac{R_p}{\sigma_p}$ (où R_p désigne le rendement du portefeuille et σ_p désigne la volatilité du portefeuille), pour inclure la volatilité dans l'optimisation et obtenir des portefeuilles plus stables et moins risqués (à risque égal), l'autre soutenait que la bonne métrique était le rendement seul R_p car, en pratique, les clients se moquent du ratio de Sharpe et si on leur montre un rendement inférieur à un autre, ils refusent d'investir, peu importe la stabilité du portefeuille. A l'issue de cette discussion, nous avons élaboré ensemble deux métriques de récompense à utiliser au choix :

— La première : $reward = R_p \cdot \sigma_p - \beta \cdot TC$, où TC représente le coût de transaction de la période et β est un paramètre arbitraire, à optimiser ;
— La seconde : $reward = R_p - \gamma \cdot \sigma_{-,p} - \beta \cdot TC$ où $\sigma_{-,p}$ est ce qu'on appelle la semi-volatilité négative, qui est défini comme l'écart-type des rendements négatifs et où

γ est un nouveau paramètre arbitraire à optimiser.

On voit déjà que le nombre de paramètres à optimiser commence à s'agrandir et que leur espace est continu. Puis, une fois l'environnement, l'agent et la récompense définis, il faut construire par dessus la *politique* de décision.

rl policy Ce programme contient une classe intitulée Policy qui dicte le processus de décision de l'algorithme. Elle est clef car c'est elle qui est optimisée au cours du temps, qui est facteur ou non de performance, qui donne la marche à suivre pour apprendre de l'environnement, etc. Cette "politique" est à choisir parmi de nombreuses disponibles dans la littérature de reinforcement learning. Nous avons dû choisir un point de départ et après discussion et lecture de la littérature scientifique, nous avons décidé de nous tourner vers les méthodes dites "*actor-critic*", qui combienent les logique de *policy method* et de *value-method*.

Il est important, à ce stade, de revenir à la structure de notre problèmes et des données qui sont disponibles. Les premiers exemples notoires de reinforcement learning ont concerné des jeux simples et les espaces d'états comme d'actions étaient discrets (dans un jeu comme Pong, la raquette peut aller en haut ou en bas, d'un ou plusieurs niveaux, tandis que les états observés sont un nombre de pixels colorés en noir ou en blanc, ou encore la position sur une grille de la balle, etc.). Le fait que ces espaces soient discrets rend possible une cartographie précise et bijective de l'espace états-actions-récompenses ; dans un problème comme le nôtre, l'espace d'états est continu (des variables réelles standardisées) et l'espace d'actions, bien que discret au départ (stratégie *long-flat* donc poids binaires $0/1$), a vocation à être continu (des poids d'actifs dans un portefeuille). Il faut donc prendre en compte le fait que cette cartographie bijective d'états-actions-récompenses n'est pas tractable numériquement : la solution envisagée le plus fréquemment est l'utilisation des réseaux de neurones, c'est par conséquent la solution que nous avons choisie.

Cette classe contient plusieurs méthodes : la première est, comme à l'accoutumée, une classe d'initialisation et cette dernière est construite par dessus l'écosystème PyTorch, connu pour sa facilité d'utilisation et sa performance computationnelle. Parmi les paramètres de cette initiatialisation, on trouve la structure du réseau de neurones, son nombre de couches cachées, son nombre de neurones, leur interconnexion, le solveur utilisé, le taux d'apprentissage, ...
Ensuite, cette classe contient une méthode qui prend en entrée un état de l'environnement

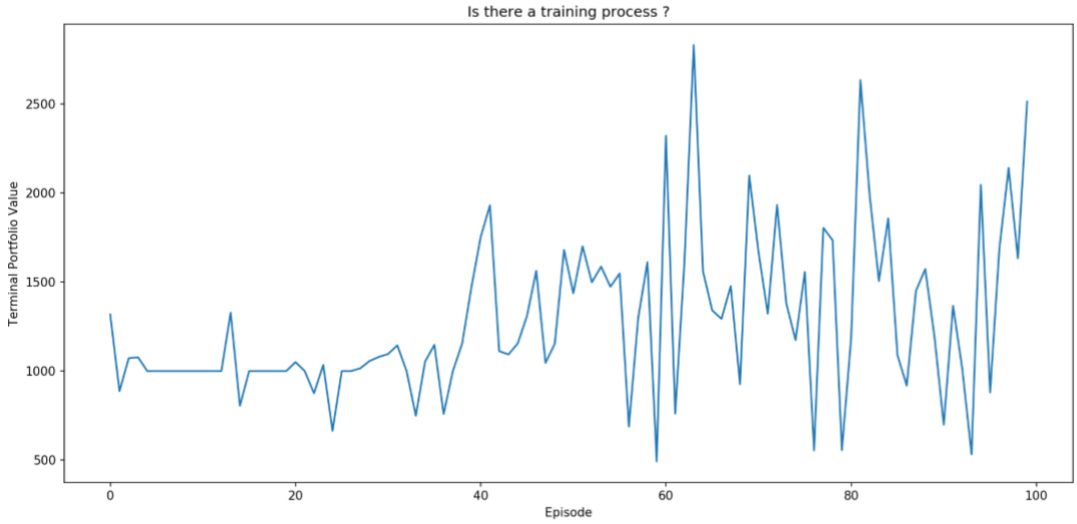
observé et renvoie, grâce au réseau de neurones précédemment défini, des probabilités de choisir l'une ou l'autre des actions disponibles.
Puis, elle contient une méthode qui stipule la manière de traiter ces probabilités et de les convertir en action (tirage aléatoire selon les probabilités, choix de l'action la plus probable, ...).
Enfin, elle contient une méthode qui actualise le réseau de neurones en fonction des récompenses observées, de la manière de traiter ces dernières (actualisation) et de la fréquence à laquelle les traiter.

launch train Une fois les deux classes définies ci-dessus instanciées, tout est prêt pour lancer un test et voir si un agent peut effectivement avoir un comportement qui nous intéresse. Ce programme vise, par dessus les deux précédents, à entraîner un agent et à observer son comportement et son apprentissage. Il ne constitue pas en soi de base à une optimisation mais permet, pour une exploration, d'observer de près ce que fait l'algorithme et de modifier manuellement ses paramètres.

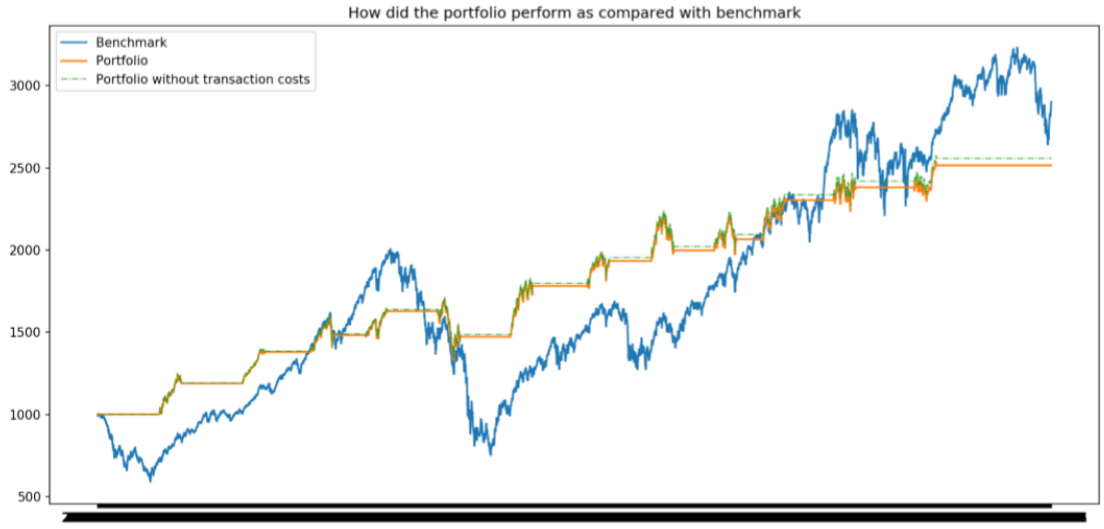
La première partie de ce programme est calculatoire. Une méthode permet de définir, à partir des deux méthodes précédentes, un environnement, un agent, un système de récompense, une politique de décision et d'entraîner l'agent. Encore une fois, de nombreux paramètres entrent en compte : parmi eux, le plus important à citer est la fréquence de la mise à jour de l'apprentissage, associée aux nombres de *replays*. Ces derniers sont une spécificité de l'architecture reinforcement learning. En effet, l'idée est que séquentiellement, à une fréquence donnée, l'agent regarde son passé proche et adapte ses prédicteurs en fonction de ce qu'il a fait, de ce qu'il avait observé pour agir ainsi, de la manière dont il a été récompensé, etc ; de manière conjointe, l'agent, comme un joueur qui rejouerait des milliers de partie pour s'améliorer progressivement, peut rejouer son passé et essayer de faire mieux. On peut donc faire varier à la fois la taille de l'historique (en multipliant ce dernier à l'envie) et le nombre de mises à jour des prédicteurs.

Le second volet de ce programme, tout aussi important dans une dynamique d'exploration des potentialités du reinforcement learning pour la gestion de portefeuille, est constitué de visualisations. Afin de voir rapidement si un ensemble de paramètres choisis pouvait entraîner un agent à battre un indice de référence sur une longue période (avec en ligne de mire des *backtests* donc), j'ai construit un rapport pdf généré automatiquement. Ce rapport pdf contient plusieurs graphiques. Dans ce qui suit, les graphiques exposés sont issus du même rapport, donc d'un même ensemble de paramètres.

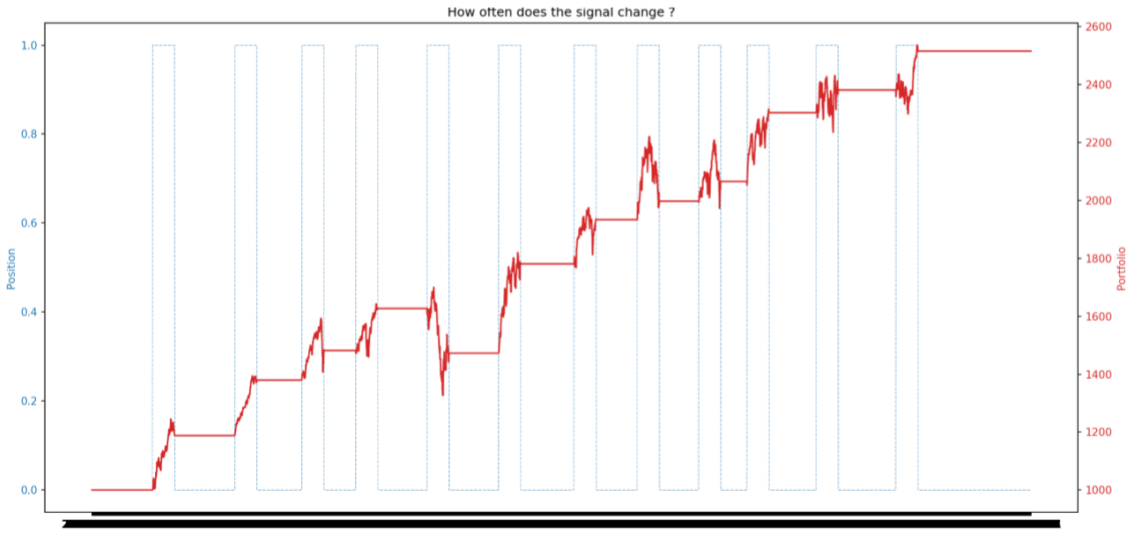
1. Le premier graphique donne un indice visuel sur l'existence d'un processus d'apprentissage. Il donne la valeur finale du portefeuille détenu par l'agent après apprentissage, en fonction de l'épisode (un épisode étant défini comme le passage d'un historique complet ; on revient au point de départ après chaque épisode, mais les prédicteurs sont mis à jour au fur et à mesure). L'idée est que si l'algorithme fait ce qu'on attend de lui, plus il y a d'épisodes, plus l'agent a été performant et donc plus la valeur terminale est haute. On souhaite donc obtenir une courbe plutôt croissante, monotone si possible, et l'on ne veut surtout pas voir une série ressemblant à du bruit. Voici un exemple, relativement favorable, d'un tel graphique :



2. Le deuxième graphique montre la valeur du portefeuille construit sur le dernier épisode. L'idée est de voir, parmi tous les épisodes, celui qui est supposé le plus achevé, à savoir le dernier, pour observer son comportement. Les phases de baisse ont-elles été évitées ? Les phases de hausse ont-elles été capturées ? Le *timing* de l'agent a-t-il été bon ? Autant de questions auxquelles un graphique répond plutôt mieux qu'un tableau. En voici un exemple :



3. Le troisième graphique permet de voir comment le signal de trading (la décision d'être *long* ou *flat*) se comporte. Est-il erratique ? Fait-il des allers-retours qu'on préfèrerait ignorer ? Plus simplement, cela correspond-il à ce qu'un humain éclairé ferait ? Est-ce vendable ? Le graphique montre la performance du portefeuille tout au long de l'historique en même temps qu'il décrit l'investissement en actions, entre 0 et 1.



4. Le dernier élément de ce rapport n'est pas un graphique mais rapporte un test d'hypothèse très simple. La question posée est très simple et le tableau vient compléter le premier graphique : y a-t-il apprentissage ? Ou dit autrement, la valeur terminale du portefeuille est-elle significativement plus élevée lorsque le nombre d'épisode

est grand (un épisode est un passage en revue de l'historique ; pour rappel, l'agent prend des décisions au cours de l'historique, puis à la fin de l'historique, on revient à 0 avec des prédicteurs plus précis et on commence un nouvel *épisode*). Ce qui est présenté est donc un résumé statistique d'une simple régression linéaire de la valeur terminale du portefeuille sur le numéro de l'épisode. Ce qu'on regarde alors est la "p-valeur" du numéro de l'épisode. Si cette dernière est significativement différente de 0 ($p < 0.01$), alors on considère que le numéro de l'épisode est une variable d'importance et donc qu'il y a apprentissage (si le paramètre est positif, on peut l'espérer...). Voici un exemple de ce tableau :

OLS Regression Results

Dep. Variable:	y	R-squared:	0.173
Model:	OLS	Adj. R-squared:	0.165
Method:	Least Squares	F-statistic:	20.51
Date:	Thu, 22 Aug 2019	Prob (F-statistic):	1.68e-05
Time:	09:23:35	Log-Likelihood:	-743.66
No. Observations:	100	AIC:	1491.
Df Residuals:	98	BIC:	1497.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	955.8704	82.336	11.609	0.000	792.477	1119.263
x1	6.5069	1.437	4.528	0.000	3.655	9.358

Omnibus:	9.638	Durbin-Watson:	1.957
Prob(Omnibus):	0.008	Jarque-Bera (JB):	15.027
Skew:	0.393	Prob(JB):	0.000546
Kurtosis:	4.729	Cond. No.	114.

Ces trois programmes, mis bouts à bouts, permettent de lancer des *backtests*. On appelle backtest un test qui répond à la question "qu'aurait été mon résultat aujourd'hui si j'avais mis en place cette action dans le passé ?" ; ces backtests permettent d'utiliser le passé pour optimiser, par exemple, les paramètres d'une stratégie de trading. Cela peut-être utile pour comprendre *a posteriori* si une stratégie fonctionne, pourquoi elle fonctionne, si elle comporte des écueils et lesquels, etc. Elle permet aussi de voir quels paramètres donnent de bons résultats, si ces résultats sont stables (ou relèvent du hasard), etc. En combinant tous les paramètres qu'on a cités ci-dessus, et en ajoutant ceux passés sous silence car moins importants, on obtient une petite vingtaine de paramètres à optimiser, ce qui est énorme, d'autant plus que la grande majorité de ces paramètres évoluent dans un espace continu. L'optimisation est donc très coûteuse en termes de temps et puisque la dynamique première relevait de l'exploration, j'ai mis en place la logique suivante :

1. Pour chaque paramètre, définir une fenêtre raisonnable et l'importance *a priori* du paramètre (qui va déterminer le nombre de valeurs à tester) ;
2. Pour quelques paramètres, donner des relations explicites qui vont permettre de tester des couples de paramètres et d'en éliminer d'autres. Par exemple, si l'on a un paramètre de temps long un autre de temps court, il faut toujours que le premier soit supérieur au second ;
3. Croiser tous les paramètres possibles et obtenir la "grille" sur laquelle on va "back-tester" ;
4. Calculer, à partir d'estimations préalables, le temps pris par ces opérations et définir un temps raisonnable ;
5. En déduire un paramètre d'*alea*, défini comme $\varepsilon = \frac{\text{temps disponible}}{\text{temps nécessaire}}$ qui va servir à réduire le temps. Pour chaque couple de paramètres, le backtest est lancé avec la probabilité ε.

Cette logique m'a permis de faire une sorte de tir à l'aveugle sur les couples de paramètres pour voir, en un temps raisonnable, si certains couples de paramètres donnaient des résultats intéressants.

3.2 *Trial-and-error* : l'amélioration progressive de l'algorithme, de sa structure et de son comportement

Cette série de backtests et leur visualisation m'a permis d'observer le comportement de mon algorithme et de le modifier à la marge. Parmi les graphiques en sortie, un en particulier m'a aidé à voir clair dans la stratégie de l'algorithme : c'est celui qui montre la valeur du portefeuille en même temps que la position du portefeuille (*cash* ou *equity*). Observons le graphique ci-dessous :

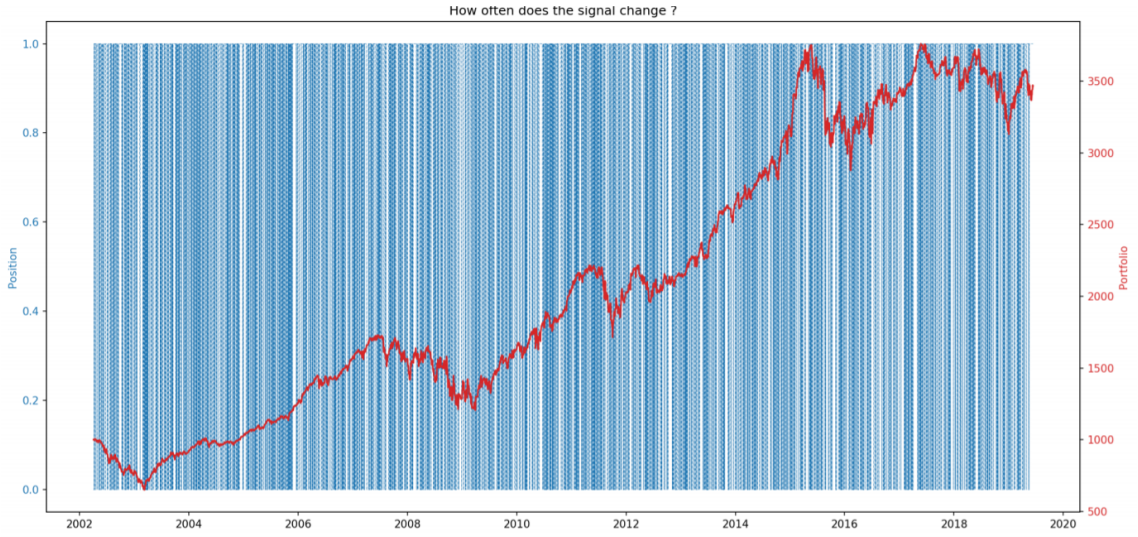


FIGURE 5 – Graphique position-performance au départ

Quelque chose frappe d'emblée : le caractère erratique de la position, oscillant chaotiquement entre 0 et 1. Outre les coûts de transaction prohibitifs [1], une telle stratégie est parfaitement invendable pour un gestionnaire de portefeuille qui raisonne sur le moyen-long terme, quand bien même elle serait profitable. Le choix qui a été fait a donc été de contraindre l'agent à conserver son portefeuille en l'état une fois un changement effectué. En d'autres termes, à chaque transaction (à la vente ou à l'achat), le portefeuille est gelé pour une durée de n jours (n devenant un paramètre à optimiser). Une fois cette modification apportée, le comportement de l'algorithme s'est modifié et le graphique ressemblait tendanciellement à celui ci-dessous :

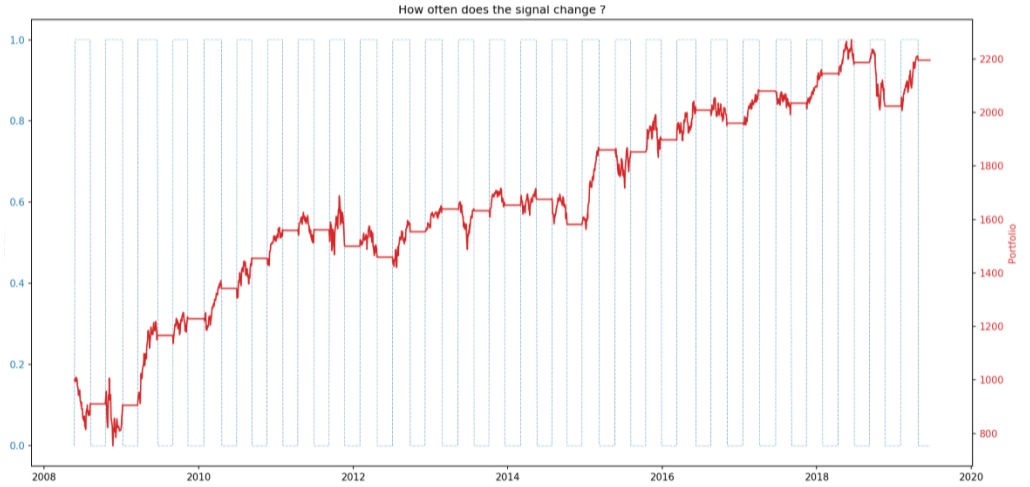


FIGURE 6 – Graphique position-performance après mise en place du gel de portefeuille

1. Avec une hypothèse raisonnable de coût de transaction - 0.1% -, il suffit de 347 entrées-sorties successives dans le portefeuille pour perdre plus de 50% du capital initial. En effet, $(1-0.001)^{2 \cdot 347} = 0.499$.

Ce graphique pose encore problème. En effet, au-delà de la forme séduisante qu'a la courbe de performance, quasi monotone à la hausse, la position oscille toujours entre 0 et 1, certes plus lentement. Une inspection du code, plus précisément de la politique de décision, montre que, pour garder une flexibilité entre exploration et exploitation (les deux piliers du reinforcement learning), le processus de décision marche comme suit : le réseau de neurones donne des probabilités d'être à 0 ou 1 (en *cash* ou en *equity*) et la position est tirée *aléatoirement* selon ces probabilités. Cette logique est intéressante lorsque les probabilités sont très discriminantes. En effet, si la distribution est de l'ordre de 90%/10%, on est à peu près sûr que le choix prépondérant va être fait, sauf de temps en temps (et c'est là où l'exploration aura lieu) ; et de manière générale, l'idée est que plus le temps passe, plus les probabilités sont discriminantes et donc plus l'exploitation prend le pas sur l'exploration. Cependant, sur les marchés financiers, il est extrêmement rare d'avoir un signal certain à 80%. Au contraire, un signal précis à 60%, voire à 55%, suffit pour être profitable sur le long terme. Mais dans notre logique, cela signifie que les probabilités tournent entre 60%-40% et 51%-49%, ce qui conduit à une oscillation systématique.

J'ai donc changé le processus pour avoir quelque chose de plus stable mais surtout de plus conforme au comportement que nous recherchions. Le processus mis en place a été le suivant :

1. Avec une probabilité ε, choisir une action au hasard (exploration) ;
2. Avec une probabilité $1 - \varepsilon$, utiliser le réseau de neurones et obtenir les probabilités d'être dans chacune des positions (exploitation) ;
3. Calculer la différence des probabilités et la comparer à un seuil appelé "seuil d'indécision" noté s_{ind}.
4. Si cette différence est supérieur à ce seuil, prendre la décision préconisée par les probabilités (prendre la position avec la plus grande probabilité ; on voit ici la probabilité comme un poids) ; sinon, garder la position de la période précédente.

Cette logique permet de garder la dichotomie entre exploitation et exploration en quittant le côté chaotique du signal. ε et s_{ind} deviennent, encore une fois, des paramètres à optimiser. Cette seconde modification d'importance conduit à un graphique ressemblant cette fois au graphique ci-dessous :

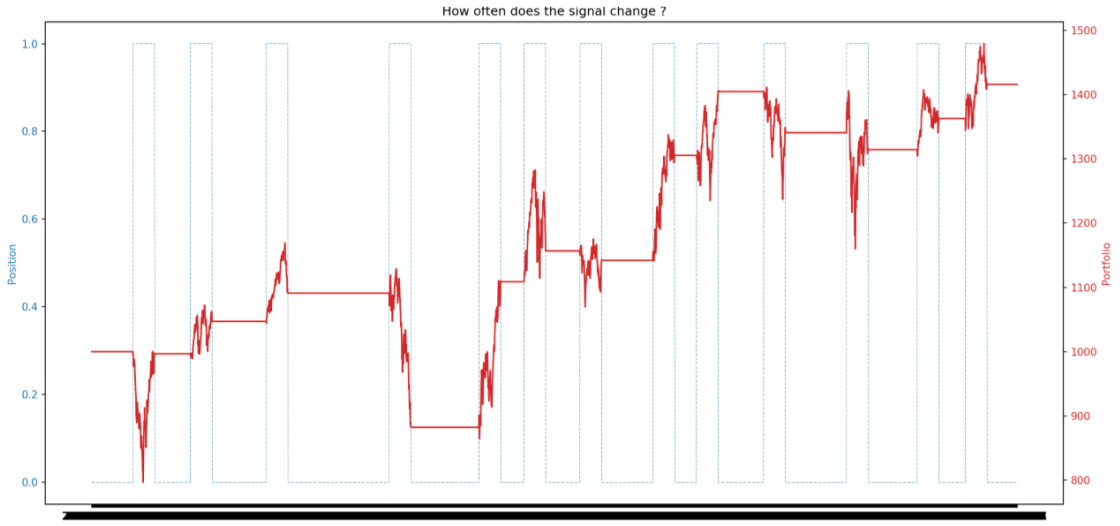


FIGURE 7 – Graphique position-performance après le changement du processus de décision

On voit désormais qu'il y a des périodes d'intermittence avec plus de transactions et que d'autres périodes sont plus calmes. Le comportement ressemble déjà plus à un comportement que l'on souhaiterait dans un fonds du type d'Aequam Capital.

3.3 Analyse des résultats

Pendant plusieurs semaines, les backtests que j'ai lancés ont tourné sur mon ordinateur en toile de fond, la nuit, les week-ends etc. Force est de constater que les résultats ont été relativement minces et que rares sont les agents entraînés par mon algorithme qui ont surperformé l'indice de référence. Après analyse des graphiques, du code et du problème, voici les causes potentielles de cet échec :

Absence de signal dans les données Il est clair, d'après le bon sens mathématique mais surtout d'après la littérature, que les coûts de transaction effacent les maigres performances d'un trading basé sur un signal faible [2]. Pour battre un indice de référence en incluant les coûts de transaction, il faut donc avoir une stratégie systématique fondée sur un signal fort. L'absence de résultats tangibles peut être dûe à l'absence de signal dans les données. Comme on l'a dit, l'univers de séries temporelles corrélées à la performance à prédire est très vaste et si l'on ajoute toutes les transformations possibles (différenciation, passage en logarithme, indicateurs techniques, ...), il devient impossible d'être exhaustif.

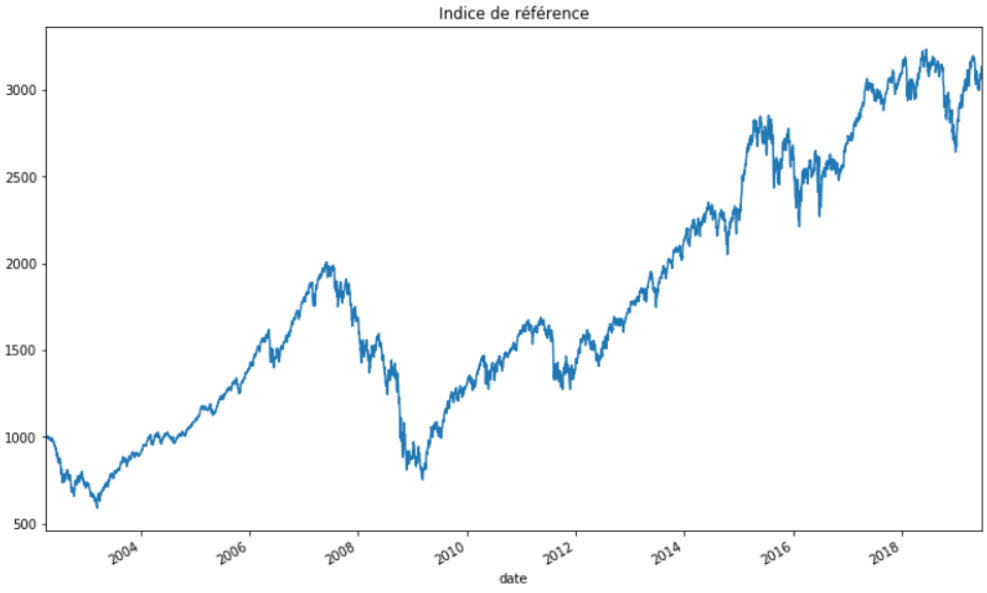
2. Voir le développement du livre *Trading and Exchanges* - [20] - sur la dichotomie trader informé - trader non informé.

Les 7 jeux de données choisis sont donc des exemples de données construites à partir de l'expertise de mes encadrants, mais aucun résultat théorique n'assure qu'un signal tangible soit systématiquement exploitable. En outre, il n'est même pas sûr qu'il soit possible de prédire à un jour les cours des facteurs en question. Mais entrer dans cette question frapperait de nullité le travail effectué et l'essence de ce que vendent les fonds d'investissements est une capacité à voir ce que d'autres ne voient pas, et donc d'une certaine manière à être plus capables que d'autres de "prédire" la performance des facteurs.

Un problème avec peu de solutions envisageables Il faut ici revenir sur la définition précise de notre première étape, à savoir écrire un algorithme qui puisse entraîner un agent de trading capable de surperformer un indice de référence sur une période donnée. Les conditions précises sont les suivantes :

1. L'agent de trading n'a que deux positions possible, investi ou non investi ; sa performance est donc celle de la référence à battre quand il est investi, ou une performance nulle quand il n'est pas investi. Le comportement idéal est donc un agent qui est investi quand l'indice monte et non investi quand le marché baisse ;
2. L'agent de trading supporte des frais de transaction de l'ordre de 0.1% ;
3. L'agent de trading est vendable par une société de gestion orientée moyen terme, donc sa fréquence de trading doit être relativement basse (quelques-uns par mois au maximum).

L'indice de référence est représenté ci-dessous, sur la période 2003-2019 :



L'indice est très haussier sur la période, le comportement idéal qu'on souhaiterait avoir est donc : être investi tout le temps et être en cash entre fin 2007 et fin 2009 et entre fin 2015 et début 2017. Eventuellement, on peut etre en cash autour de fin 2011-début 2012. C'est l'unique solution qu'on voudrait voir apparaître, tellement le marché est haussier sur la période.

Problème d'optimisation lié à la fonction de récompense On l'a dit, la fonction de récompense est un point sensible de la stratégie de reinforcement learning. C'est elle qui dicte l'adaptation de l'agent à son environnement et à ses actions passées. Or, voici le problème :

— Parmi les mesures citées au début, impliquant le rendement, la variance, la semi-variance etc, si le portefeuille est en cash, alors la récompense est de 0. Cela pose problème car dans la descente de gradients de l'optimisation, le pas induit va être de 0 et donc l'adaptation va être très lente, voire inexistante ;

— Au contraire, si l'on renverse le problème et qu'on décide de choisir le rendement relatif... c'est l'investissement dans l'indice de référence qui reçoit une récompense de 0.

Ce qui pose question est donc le fait d'avoir binarisé le problème (à partir de la question originelle, plus générale, de choix de poids de facteurs dans le portefeuille).

3.4 Etat des lieux et pistes d'amélioration

Les rendements financiers sont extrêmement difficiles à prédire. Tous les gérants prétendent pouvoir les maîtriser mais en dehors de quelques marchés de niche excessivement déséquilibrés (le marché des *dividend swaps* par exemple), ils sont très aléatoires et les potentiels signaux sont très bruités. Par conséquent, face à l'échec de l'algorithme mis en place, une question a beaucoup occupé mon esprit : l'algorithme ne trouve-t-il rien parce que les données ne donnent pas de signal tangible ? Ou ne trouve-t-il rien parce que je l'ai mal écrit ? Etant dans une dynamique d'exploration au sein d'Aequam Capital, encadrés par des personnes peu ou prou dans la même situation que moi, la réponse est loin d'être évidente.

Soucieux de quitter Aequam Capital sans rester sur un échec, soucieux également de faire de mon mieux et de ne pas m'obstiner dans une voie de mauvais augure, j'ai suggéré d'opérer un pivot, de changer de prisme. Après discussion, deux voies ont été envisagées :

Remise en cause de l'algorithme L'algorithme ne trouve-t-il rien parce que je l'ai mal écrit ? Si la réponse à cette question est positive, alors il faut tout de même voir si le reinforcement learning peut apporter quelque chose au problème de la gestion de portefeuille. La piste qui a été choisie a donc été de prendre des algorithmes tout faits, par exemple ceux disponibles sur Open AI Gym, et de les adapter à notre problème *sans les modifier*. Cette opération, impossible au début car j'étais trop néophyte pour comprendre les librairies en profondeur et avais besoin de plus d'étapes intermédiaires de visualisation, devient possible à la fin de la mission et est salvatrice car ces algorithmes ont des performance éprouvées sur de nombreux problèmes (ce qui n'est pas le cas du mien).

Amélioration de la fonction de récompense Si l'on regarde la sous-partie précédente, on voit que la binarisation du problème (*long-flat* sur un indice de référence) pose problème à deux égards : elle rend le problème extrêmement contraint et elle engendre une fonction de récompense nulle sur toute une partie de l'espace états-actions. Une solution envisagée a été de changer l'espace d'actions : plutôt que d'avoir seulement comme possibilité *cash* ou référence, il a été imaginé, en conformité avec ce qui se fait déjà chez Aequam Capital, d'étendre l'espace d'états à "*cash*, défensif, référence, offensif", passant donc de 2 à 4 états disponibles. Par construction, une fonction de récompense construite à partir de la performance relative pourrait donc donner des récompenses négatives ou positives selon les moments, et moins souvent nulles. Cependant, cette solution, construite en toile de fond pendant un moment, s'avère peu ambitieuse et peu différente de la démarche déjà envisagée. Il a donc été décidé, dans une logique toujours d'exploration et d'essai, d'explorer une autre voie, celle de l'Inverse Reinforcement Learning (IRL). On l'a vu, la fonction de récompense pose problème, on la modifie à la main, voire on l'optimise. De fait, la logique ludique atteint vite ses limites car il est très difficile, dans grand nombre de problèmes concrets, de créer une fonction de récompense qui capture parfaitement le comportement que l'on souhaite observer. L'inverse reinforcement learning tente donc de renverser le problème et, à partir de comportements supposés optimaux, d'en déduire la fonction de récompense. Trois articles m'ont permis de comprendre les enjeux de l'inverse reinforcement learning : *Algorithms for Inverse Reinforcement Learning* (Ng et Russell, [21]) pour une introduction, *Maximum Entropy Inverse Reinforcement Learning in Continuous State Spaces with Path Integrals* (Aghasadeghi et Bretl, [22]) et *Model-Free Deep Inverse Reinforcement Learning by Logistic Regression* (Uchibe, [23]) pour des articles plus spécifiques à des espaces d'états continus.

Au moment où ce rapport doit être rendu, c'est-à-dire un mois avant la fin de ma mission chez Aequam Capital, ce sont ces deux branches - adaptation de librairies et

algorithmes éprouvés à notre problème et exploration de l'inverse reinforcement learning - qui sont à l'étude, pour tenter de laisser derrière moi quelque chose de réutilisable chez Aequam Capital.

4 Conclusion

Il est difficile de conclure sur un projet qui n'est pas terminé et qui en est à sa phase d'ouverture maximale. L'étude de la littérature du reinforcement learning provoque un sentiment de vertige quand on la confronte à l'idée que la mission porte sur à peine plus de trois mois. En effet, le nombre d'algorithmes disponibles, les idées qui les sous-tendent, le peu d'application tangibles et éprouvées ouvre un champ vaste à l'imagination de celui qui est curieux. Pendant ces quelques mois, une piste a été explorée, avec peu de succès, mais d'autres pistes existent ; il est difficile de savoir si les bons choix ont été faits tant j'ai découvert le champ du reinforcement learning en même temps que ceux qui m'encadraient.

Indépendamment des résultats concrets, cette mission a été l'occasion pour Aequam Capital, société de gestion quantitative en développement, de s'exposer à des méthodes jusque là inconnues, de se confronter à des logiques peu familières (d'investissement aléatoire, de modèle difficile à interpréter, etc), d'apprendre sur les ressources et les limites de ce qu'on appelle parfois un peu vite le "Machine-Learning" ; en contrepartie, cette mission a été l'occasion pour moi de confronter mes connaissances théoriques, mes modèles avec données distribuées aléatoirement, à la réalité pratique des données bruitées, trop nombreuses, trop autocorrélées. Elle a donc été pour toutes les parties prenantes une source d'apprentissage et de stimulation intellectuelle : j'ai personnellement beaucoup appris sur le reinforcement learning, mais aussi sur l'industrie de la gestion de portefeuille et sur ses réalités quotidiennes.

Comme tout projet de recherche, ce projet a donné lieu à des frustrations. Frustration de ne pas obtenir de résultats, frustration de passer beaucoup d'heures à mettre en pratique une idée supposée astucieuse sans voir bouger le résultat d'un pouce, frustration d'écrire un rapport dans un sentiment d'inachevé. Mais il a surtout donné lieu à des moments de jubilation, à des périodes de grande curiosité comblée par la littérature scientifique passionnante, à des moments de questionnements sur la nature de la logique algorithmique, sur le fonctionnement ou sur les performances exceptionnelles de certains algorithmes. Il est à espérer que les dernières semaines seront propices à l'inspiration intellectuelle et que de cette mission découleront une compétence et un apprentissage sur lesquels capitaliser.

5 Remerciements

Je souhaite remercier mon manager chez Avisia, Matthieu Pourbaix, qui m'a accompagné pendant toute mon alternance, sur cette mission et sur d'autres. Sa façon d'encadrer, entre confiance et exigence, a été précieuse pour moi et a beaucoup contribué à nourrir mon envie de rendre un travail bien fait et rigoureux. Son regard critique m'a permis, à des moments clefs de mes missions, d'infléchir une partie du travail fourni pour ne pas continuer dans une voie sans issue. Ses qualités humaines ont fini de me convaincre qu'il est presque aussi important de choisir les personnes avec qui on travaille que le sujet sur lequel on travaille.

Je souhaite ensuite remercier les gérants (respectivement CIO et PM) d'Aequam Capital, Thierry Béchu et Pierre Colonna, qui m'ont encadré pendant toute la durée de cette mission. Leur façon de susciter l'échange, leur éclairage sur des points précis de finance ou d'analyse technique, leur implication dans le projet ont contribué à maintenir une implication et une motivation constante pour un projet par essence soumis à l'*alea* des résultats.

Je souhaite encore remercier Florian Berg (chercheur au MIT employé à temps partiel par Aequam Capital) pour sa direction lointaine mais efficace. Il a su me donner des directives assez précises pour qu'à aucun moment je ne me sente abandonné à moi-même sur un sujet sur lequel je construisais ma connaissance au fur et à mesure. A des moments décisifs, il a su me donner des éclairages techniques et des indications précieuses sur la marche à suivre.

Je souhaite enfin remercier mes collègues chez Avisia et chez Aequam Capital, dont la bonne humeur et l'accueil ont été une source de motivation permanente au cours de mon alternance en général et de cette mission en particulier.

Références

Introduction au reinforcement learning

[1] Mnih et al., *Playing Atari with Deep Reinforcement Learning*, 2017, DeepMind

[2] TowardsDataScience.com, *Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG)*

[3] LearnDataSci.com, *Reinforcement Q-Learning from Scratch in Python with OpenAI Gym*

[4] Csaba Szepesvari, *Algorithms for Reinforcement Learning*, 2009, Morgan & Claypool Publishers

[5] Tal Perry for Medium.com, *I Went To The German Alps and Applied Reinforcement Learning To Financial Portfolio Optimization*

[6] Gordon Ritter, *The Usefulness of Reinforcement Learning in Finance*, 2018, New York University, Rutgers University, Baruch College

[7] Skymind.ai, *A Beginner's Guide to Deep Reinforcement Learning*

[8] Louiskirsch.com, *A Map of Reinforcement Learning*

[9] Nicolas Baskiotis, *Apprentissage par renforcement*, 2018, Laboratoire d'Informatique de Paris VI (LIP6), Sorbonne Université

[10] Andrej Karpathy blog, *Deep Reinforcement Learning: Pong from Pixels*

Le reinforcement learning appliqué à la finance

[11] Nevmyvaka et al., *Reinforcement Learning for Optimized Trade Execution*, 2006, Lehmann Brothers, University of Pennsylvania

[12] Moody et al., *Learning to trade via Direct Reinforcement*, 2001, IEEE Transactions on Neural Networks and Learning Systems, vol 12

[13] Matsui et al., *Compound Reinforcement Learning : Theory and An Application to Finance*, 2011, Chubu University, Bank of Tokyo-Mitsubishi UFJ, University of Tokyo, Presto JST

[14] Deng et al., *Deep Direct Reinforcement Learning for Financial Signal Representation and Trading*, 2016, IEEE Transactions on Neural Networks and Learning Systems

[15] Dempster et al., *Intraday FX Trading : An Evolutionary Reinforcement Learning Approach*, 2002, Research Papers in Management Studies, University of Cambridge

[16] Dempster et al., *An automated FX trading system using adaptative reinforcement learning*, 2004, Research Papers in Management Studies, University of Cambridge

[17] Bates et al., *Evolutionary reinforcement learning in FX order book and order flow analysis*, 2003, Centre for Financial Research, Judge Institute of Management, University of Cambridge

[18] Jiang et al., *Cryptocurrency portfolio management with deep reinforcement learning*, 2017, Xi'an Jiaotong-Liverpool University

[19] Jiang et al., *A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem*, 2017, Xi'an Jiaotong-Liverpool University

Autres

[20] Larry Harris, *Trading and Exchanges : Market Microstructure for Practitioners*, 2002, OUP USA

[21] Ng et al., *Algorithms for Inverse Reinforcement Learning*, 2000, Berkeley

[22] Aghasadeghi et al., *Maximum Entropy Inverse Reinforcement Learning in Continuous State Spaces with Path Integrals*, 2011, IEEE/RSJ

[23] Uchibe, *Model-Free Deep Inverse Reinforcement Learning by Logistic Regression*, 2017, Springer

Note de synthèse - Reinforcement Learning appliqué à la gestion de portefeuille chez Aequam Capital

Dorian LAGADEC

Année scolaire 2018-2019

L'organisme : Aequam Capital

Aequam Capital est une société de gestion quantitative employant une dizaine de personnes et spécialisée dans la gestion multi-facteurs. Le fonds Aequam Dynamic Premia Equity (ADPE), ouvert il y a un an, a une allocation d'actifs pilotée par un algorithme propriétaire.
Dans le cadre de la recherche d'Aequam Capital, il a été décidé d'explorer des pistes jusqu'alors inconnues dans le but de générer une performance supérieure à celle connue actuellement. Une piste envisagée est celle du Reinforcement Learning - ou apprentissage par renforcement - car c'est un cadre théorique qui permet de répondre à des problèmes séquentiels et d'avoir une dynamique double, d'exploration et d'exploitation. C'est la raison pour laquelle Aequam Capital a fait appel à la société Avisia, qui m'a accompagné au cours de mon alternance, pour proposer mon travail : ayant un pied dans le monde académique et de la recherche avec l'ENSAE et un autre dans la finance de marché avec mes précédents stages, j'étais susceptible de les aider dans leur processus de découverte.
Cette mission d'exploration a été menée de la manière suivante : j'étais responsable de lire la littérature scientifique, de proposer une démarche de *Proof-Of-Concept* (POC) et de la mettre en oeuvre ; les gérants d'Aequam Capital Thierry Béchu et Pierre Colonna m'apportaient leur éclairage sur des questions spécifiques de finance de marché, de données et de considérations pratiques sur le caractère faisable ou non des allocations algorithmiques proposées ; le chercheur avec lequel travaille Aequam Capital, Florian Berg, était chargé de m'aider sur l'aspect théorique et fonctionnel des algorithmes utilisés.

Un problème précis et une démarche incrémentale

Le problème était le suivant : étant donné 6 facteurs de risques (un facteur de risque étant un panier pondéré d'actions exposées à un même risque donné), est-il possible

de choisir dynamiquement les poids de ces facteurs pour que la valeur finale du portefeuille soit maximisée (sous contrôle de critères de volatilité et de *turnover*), en prenant en compte des coûts de transaction ? Si oui, l'allocation dynamique proposée est-elle interprétable, est-elle vendable ?
Le problème étant relativement ouvert, il a été décidé de commencer par un problème plus simple : l'indice de référence étant un portefeuille équi-pondéré de ces 6 facteurs, est-il possible de choisir des points d'entrée et de sortie optimaux dans ce portefeuille équi-pondéré pour battre ce portefeuille ? Ce problème est plus simple car plus contraint : les poids sont tous égaux à $\frac{1}{6}$ ou à 0.
Etant néophyte dans le domaine du Reinforcement Learning, la démarche a donc été la suivante : familiarisation avec les algorithmes de Reinforcement Learning, puis lecture d'articles qui portent sur l'application du Reinforcement Learning à des problématiques financières, enfin exploration de librairies en ligne mettant en oeuvre lesdits algorithmes. Après cette phase de découverte, j'ai codé en Python mes propres classes pour adapter un algorithme de Reinforcement Learning au problème d'Aequam Capital. A l'aide d'outils de visualisation, j'ai pu observer le comportement de l'algorithme entraîné sur le jeu de données et, à plusieurs reprises, j'ai modifié tel ou tel bout de code pour que l'algorithme modifie son comportement afin que ce dernier s'approche au plus près d'une allocation dynamique "logique", du goût des gérants.

Etat des lieux à date et pistes d'amélioration

Après de nombreux *backtests*, aucune instance de mon algorithme n'a pu significativement faire mieux que l'indice de référence et j'ai tenté d'analyser les causes de cet échec. D'abord, il est possible, de nombreux articles le disent, qu'il soit impossible de prédire les cours boursiers, donc qu'aucun signal ne puisse être détecté dans les données. Si l'on écarte cette hypothèse qui ruine la raison d'être de ce projet, on peut soupçonner les données choisies et/ou leur traitement de ne pas donner lieu à la détection d'un signal fort pour notre algorithme. Autrement, il est possible que l'algorithme que j'ai codé soit largement sous-optimal, c'est pourquoi j'ai décidé de revenir aux algorithmes éprouvés (ceux d'Open AI Gym par exemple) pour voir si le problème venait de mon algorithme ou de mes données. Enfin, la question de la fonction de "récompense", clef dans tout algorithme de Reinforcement Learning car c'est elle qui indique à l'algorithme si chaque décision a été bonne ou non et à quel point, est sensible et il a été décidé d'aborder la question de l'Inverse Reinforcement Learning, une forme de *reverse engineering* du Reinforcement Learning, pour comprendre plus finement les paramètres de l'allocation que recherche réellement Aequam Capital, plus complexe qu'un arbitrage entre rendement et risque.

Summary Note - Reinforcement Learning applied to portfolio allocation at Aequam Capital

Dorian LAGADEC

School year 2018-2019

The company : Aequam Capital

Aequam Capital is a quantitative portfolio management company that employs a dozen of people and that is specialized in mutli-factor strategies. The fund Aequam Dynamic Premia Equity (ADPE), opened one year ago, is allocated among risk factors according to a proprietary algorithm..

Aequam Capital's research team decided to explore unknown ways to generate better returns. Among them is Reinforcement Learning, a framework that addresses dynamic problems and allows a part of exploration within the statistical learning strategy. This is why Aequam Capital called Avisia, the company I have worked for for the past 12 months : being a student at ENSAE dealing with academic issues and having former experiences in the financial markets' field, I was prone to helping them in their discovery project.

This exploration mission was built according to the following scheme : I was in charge of reading the scientific literature, of suggesting a proof-of-concept (POC) approach, of executing it ; Aequam Capital's managers Thierry Béchu and Pierre Colonna brought me their expertise on finance-related questions, on data issues and on practical questions concerning the feasability of suggested algorithmic allocations ; the researcher with whom Aequam Capital works, Florian Berg, was responsible for providing a kind of scientific support.

A precise problem and an incremental approach

Aequam's problem is posed as such : given 6 risk factors (a risk factor is a weighted basket of stocks exposed to the same given risk), is it possible to dynamically set these risk factors' weights in the portfolio in order to maximize (subject to volatility and portfolio turnover constraints) the terminal value of the portfolio, once we include actual transaction costs ? If so, can the dynamic allocation be explained ? Can it be sold to actual investors ?

The problem being relatively open, it was decided to start with an easier problem : the

benchmark being an equally-weighted portfolio of this 6 risk factors, is it possible to choose entry points to and exit points from this portfolio in order to outperform it ? This problem is easier because further constrained : the weights all equal $\frac{1}{6}$ or 0.
Being an amateur in Reinforcement Learning, the following approach was decided : first, I became familiar with the broad scientific literature on reinforcement learning ; then I read specific articles on its application to financial markets-related problem ; last I explored all public pieces of codes online. After this learning process, I coded in Python my own libraries to adapt a reinforcement learning algorithm to Aequam Capital's problem. Using visualization tools, I could observe the behaviour of the algorithm trained on the dataset and, a few times, I modified some parts of my code so that the algorithm should change its behaviour and stick to a so-called "logical" dynamic allocation, meaning an allocation that can be sold with a decent story.

Actual state and room for improvement

After loads of backtests, no instance of my algorithm could significantly outperform the benchmark index and I tried to analyze the reasons of this failure. First, it can be, lots of paper defend this idea, that it is not possible to predict the stocks' returns, so that no signal can be extracted from the data. If we put aside this hypothesis that ruin the very essence of this project, one can at least doubt that the chosen data, the chosen transformation of the data cannot lead to the extraction of a strong trading signal. Furthermore, it might be that the algorithm I wrote is wildy under-optmial, this is why I decided to come back to proven algorithms (those published by Open AI Gym for instance) to understand if the failure came from my algorithm or my data. Last comes the reward function, that is highly sensitive : this function is key in every reinforcement learning algorithm because it tells the algorithm whether the chosen action was good or bad and to what extent. It was decided to look at Inverse Reinforcement Learning, a kind of reverse engineering of reinforcement learning, to understand with more precision the hidden parameters behind the allocation actually wanted by Aequam Capital, way more complex than an arbitrage between risk and reward.