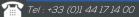
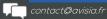


Compte-rendu: Reinforcement Learning appliqué à l'allocation de portefeuille *30 septembre 2019*







Pour comprendre : quelques notions de Machine Learning



- Machine learning vs Statistiques: plus récent, moins de théorie, plus d'empirique, moins d'interprétabilité, plus de performance
- Ligne de fracture :
 - Apprentissage supervisé: on prédit un label à partir d'exemples (iris, scoring crédit, ...)
 - Apprentissage non supervisé : on classifie des individus sans exemple (segmentation client, régime de marché, ...)

Prédiction ou classification ponctuelle. Application en finance : prédiction de rendement, day-trading, ... mal adapté

- → Pas adapté aux problèmes séquentiels
- Reinforcement Learning: utilise ML pour optimiser la trajectoire d'un agent dans un environnement donné
 - Différence avec théorie du contrôle optimale (modèles économiques) : pas d'a priori sur le comportement de l'agent
 - Puissant car repose sur le ML: pas d'hypothèse sur les données ou sur le signal à chercher, ce sont les algorithmes qui les déterminent automatiquement
 - Adapté à Aeguam Capital (dynamique temporelle, turnover, stabilité du rendement, plutôt que trading opportuniste)



Le Reinforcement Learning en quelques mots



- Sujet à la mode depuis Playing Atari with Deep Reinforcement Learning, par les ingénieurs de Google DeepMind (2013)
 - Alpha Go, vainqueur du champion de Go Lee Sedol (4-1) a achevé de prouver le succès des méthodes de RL
- **Vidéo**: DeepMind joue à Atari Breakout https://www.youtube.com/watch?v=V1eYniJORnk
 - Bonus : voir un modèle de Deep RL jouer au Snake https://www.youtube.com/watch?v=zlkBYwdkuTk
- Principe de récompense : entraîner l'ordinateur à base de récompenses ou punitions pour qu'il s'améliore graduellement
- Excellente introduction : cours de Nicolas Baskiotis (partie I) http://dac.lip6.fr/master/wp-content/uploads/2019/01/ARF-2018-cours8.pdf
 - Equation de Bellman peut avoir une solution exacte, mais en général pas suffisamment d'hypothèses \rightarrow RL





Notre problématique chez Aequam Capital



- Stratégie d'investissement *long-only*, sur des actions européennes, multi-factorielle (facteurs de risque)
- Pourquoi le RL?
 - Une logique **intuitive** et séduisante (dynamique, comportements adversariaux);
 - Une dimension **dynamique** et **exploratoire** appropriée ;
 - Un champ d'investigation vaste et **innovant** (question d'image, de disponibilité de la recherche);
 - Une littérature prolifique
- Problématique : un trader (agent) peut-il, en observant des données qui contiennent supposément un signal (environnement), choisir une allocation de portefeuille parmi plusieurs facteurs (action), afin de maximiser le rendement de long-terme sous des contraintes de volatilité et de perte maximale (récompense) ?
- Problématique simple qui cache de nombreuses contraintes opérationnelles (règlementation, turnover) et commerciales (interprétabilité, discours marketing, cohérence temporelle) → démarche incrémentale
 - 1. Exploration d'une stratégie *long-flat* sur un **portefeuille équipondéré** de facteurs (battre le benchmark en trouvant les bons points d'entrée/sortie) → difficile et coïncide avec la phase de découverte de la mission → peu de résultats ;
 - 2. Généralisation à une stratégie « factor-picking » ou « portfolio-picking » : plus libre et coïncide avec la phase de maturité de la mission → des résultats encourageants

Les outils à disposition



Matières premières : les données

- Prix des facteurs (dynamique endogène);
- Données macroéconomiques (dynamique exogène);
- Des transformations ad hoc (séries temporelles, analyse technique).

Machinerie : les librairies de code open source

- Recoder des algorithmes est long et difficile (erreurs, efficacité computationnelle)
- Beaucoup de ressources de grande qualité en ligne
- Un travail de compréhension et de vulgarisation du travail effectué : on ne crée pas d'algorithmes, on apprend à s'en servir

Le rôle du pilote

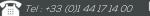
- Modéliser le système à optimiser et le traduire dans le langage du Reinforcement Learning
- Adapter les algorithmes existants pour les faire répondre à notre problème (bons algorithmes, bons paramètres, ...)
- Ecrire du code qui utilise les ressources et les codes publics pour en faire quelque chose de fonctionnel, d'utilisable et d'actionnable par la suite



Résultats



- La partie opérationnelle intéresse surtout la recherche... détaillée dans la documentation (sur Github). En quelques mots : on peut chercher longtemps avant de trouver (articles, tutos, codes mis bouts à bouts, découvertes, tentatives avortées ou infructueuses, ...)
- Résultats **encourageants** :
 - Un **code** qui tourne sans problème et a été écrit pour rester **flexible** pour les prochains utilisateurs, avec une documentation qui se veut abondante;
 - Certains backtests de très bonne qualité (Good results/2nd round);
 - Une robustesse en bonne voie de construction : du tir à l'aveugle à la recherche de zones stables pour les paramètres
 - Mais une interprétabilité quasi nulle des modèles \rightarrow système d'agrégation des modèles pour passer d'un factor-picking à une allocation (plus de stabilité, moins de coûts de transaction)





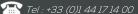
Développement et chantiers futurs



3 axes de développement :

- Recherche:
 - Feature Engineering : dépasser l'a priori sur les variables explicatives, matières premières des algorithmes
 - **Inverse Reinforcement Learning**: l'épineuse question de savoir ce qu'est une « bonne récompense »
- **Opérationnel:**
 - Nouveaux portefeuilles (flexibilité du code)
 - **Optimisation des algorithmes et hyperparamètres** (travail seulement entamé, à poursuivre)
- Phase de tests:
 - Agrégation de modèles ayant de bons backtests pour avoir une vraie allocation dynamique
 - Paper trading pour commencer à valider dans le temps les modèles obtenus



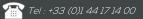






Questions





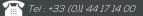




Annexes









Introduction au Reinforcement Learning



Les algorithmes principaux

Algorithm	Description	Model	Policy	Action Space	State Space	Operator
Monte Carlo	Every visit to Monte Carlo	Model-Free	Off-policy	Discrete	Discrete	Sample-means
Q-learning	State-action-reward-state	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA	State-action-reward-state-action	Model-Free	On-policy	Discrete	Discrete	Q-value
Q-learning - Lambda	State-action-reward-state with eligibility traces	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA - Lambda	State-action-reward-state-action with eligibility traces	Model-Free	On-policy	Discrete	Discrete	Q-value
DQN	Deep Q Network	Model-Free	Off-policy	Discrete	Continuous	Q-value
DDPG	Deep Deterministic Policy Gradient	Model-Free	Off-policy	Continuous	Continuous	Q-value
A3C	Asynchronous Advantage Actor-Critic Algorithm	Model-Free	On-policy	Continuous	Continuous	Advantage
NAF	Q-Learning with Normalized Advantage Functions	Model-Free	Off-policy	Continuous	Continuous	Advantage
TRPO	Trust Region Policy Optimization	Model-Free	On-policy	Continuous	Continuous	Advantage
PPO	Proximal Policy Optimization	Model-Free	On-policy	Continuous	Continuous	Advantage
TD3	Twin Delayed Deep Deterministic Policy Gradient	Model-Free	Off-policy	Continuous	Continuous	Q-value
SAC	Soft Actor-Critic	Model-Free	Off-policy	Continuous	Continuous	Advantage

Description complète: https://medium.com/@jonathan_hui/rl-reinforcement- learning-algorithms-quick-overview-6bf69736694d

Source: Wikipedia



Réalisation



Mise en place et points de vigilance

- Fonction de récompense : un calibrage primordial (équivalent du label en apprentissage supervisé)
 - Récompense = rendement
 - Récompense = rendement/volatilité
 - Récompense = rendement (a x coûts de transaction) (b x semivariance)
- Politique déterministe (dite greedy) ou probabiliste : cas particulier en finance
- Gestion de la guestion du délai : **sparse reward**, fréquence des actions différente de la fréquence des récompenses
- Qu'est-ce qu'un « bon » résultat : qu'est-ce qu'un bon apprentissage ?

© 2007-2018 AVISIA - Tous droits réservés





Réalisation



Résumé et guide pratique

- Posez votre problème en termes de RL (agent, environnement, actions, politique, récompense);
- Travaillez bien la fonction de récompense qui doit encapsuler tout le comportement recherché (penser aux effets pervers de ce qu'on récompense, les observer le cas échéant)
- **Spécifiez** les données, les travailler et les transformer
- Créez votre environnement Open Al Gym (voir l'excellente doc https://stable-baselines.readthedocs.io/en/master/index.html) : votre environnement est une classe qui doit contenir les méthodes suivantes :
 - __init__ qui spécifie l'espace d'actions (discret ou continu, bornes), d'observations (idem), la façon de l'environnement de réagir, éventuellement des paramètres supplémentaires
 - **step** qui fait passer d'une étape temporelle à une autre suite à une action
 - reset qui remet à 0 l'environnement une fois un « épisode » passé
 - render (optionnel) qui permet à l'utilisateur de visualiser un épisode
 - Ex:aequam env v1,py

Chacune de ces méthodes a des inputs et des outputs aux formats standards qu'il faut respecter pour que les algorithmes qui utilisent des environnements Gym puissent fonctionner sans problème... doc à venir chez Avisia

- Créez de quoi visualiser vos résultats, pour pouvoir améliorer graduellement votre fonction de récompense (qui ne sera pas parfaite... à explorer : Inverse Reinforcement Learning - IRL)
- **Optimisez** vos paramètres et/ou hyperparamètres (beaucoup de librairies disponibles... **scikit-optimize** par exemple)
- Comment valider un modèle : pas de validation croisée car problème de séries temporelles (données pas i.i.d)





Réalisation



Exemples de visualisation : PDP

