Gestion de portfolio par un RL *trading* BOT

Projet - Planification en Intelligence Artificielle IFT 602 / IFT 708

Présenté par :
Tahar Adam AMAIRI
Victor CHAU
Omar CHIDA
Céline ZHANG

Professeur : Froduald Kabanza
Assistants : D'Jeff Nkashama & Jordan Félicien Masakuna

Plan de présentation

I. **Introduction**
   A. Mise en situation
   B. L'état de l'art

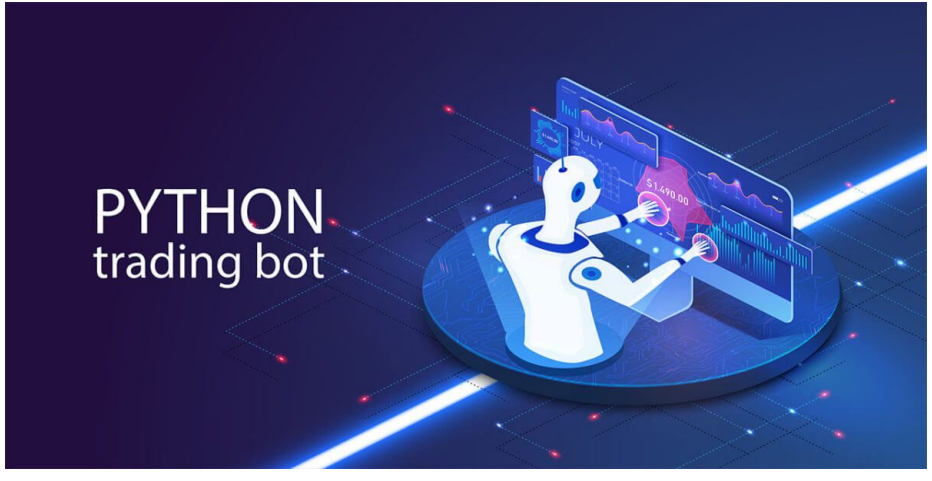
II. **Réalisation**
   A. Tâches réalisées
   B. Difficultés et amélioration

III. **Démonstration**
   A. Explication du code source
   B. Présentation des améliorations

I. Introduction - Mise en situation

Gestion d'un portfolio par un *Trading* BOT:

- Finalité: faire le plus de profit possible dans le *trade* automatisé
- Méthodes: 2 grands axes méthodologiques, ***classic trading***, ***AI trading***
- ***AI trading***: nouvelle approche, flexibilité, complexité



I. Introduction - L'état de l'art

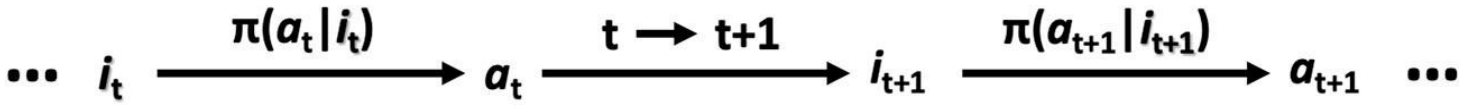
Etats: ensemble du cours de l'action sur une période

Actions: Achat (*Buy*), Vente (*Sell*), Conserver (*Hold*)

Stratégies (Politique): Stratégies de trading (*Short, Long, Buy and Hold, Sell and Hold, Trend Following et Mean Reversion*)

Récompense: Rendements journaliers

- Procédé général:
  1. Mise à jour de l'information du marché i_t
  2. Exécution de la politique pour obtenir une action π(a_t | i_t)
  3. Application de l'action de trading a_t
  4. Aller à l'étape suivante t + 1



Du DQN vers TDQN

Modifications et améliorations :

- Réseau de neurones profonds (DNN) : CNN -> DNN (*fit* les séries temporelles)
- Double DQN : 2 parties, sélection et évaluation de l'action, réduit impact de surévaluation et améliore les performances
- ADAM Optimizer : améliore la stabilité de l'entraînement et la vitesse de convergence
- Huber Loss : améliore la stabilité de l'entraînement (màj DNN est + lente et + stable)
- Gradient clipping : limite la magnitude du gradient, améliore stabilité et convergence
- Xavier initialisation : variance des gradients constante, améliore la convergence
- Batch Normalization : entraînement + rapide et + robuste, améliore la généralisation
- Régularisations : Dropout, L2, early stopping
- Pré-traitement et normalisation : filtre pour réduire bruit, transformation données
- Data augmentation : décalage du signal, filtrage de signal, ajout de bruit artificiel

Données disponibles

- Train set : 01/01/2012 - 31/12/2017 (75%)
- Test set : 01/01/2018 - 31/12/2019 (25%)

Sector	Region		
	American	***European***	***Asian***
Trading index	Dow Jones (DIA)	FTSE 100 (EZU)	Nikkei 225 (EWJ)
	S&P 500 (SPY)		
	NASDAQ (QQQ)		
Technology	Apple (AAPL)	Nokia (NOK)	Sony (6758.T)
	Google (GOOGL)	Philips (PHIA.AS)	Baidu (BIDU)
	Amazon (AMZN)	Siemens (SIE.DE)	Tencent (0700.HK)
	Facebook (FB)		Alibaba (BABA)
	Microsoft (MSFT)		
	Twitter (TWTR)		
Financial services	JPMorgan Chase (JPM)	HSBC (HSBC)	CCB (0939.HK)
Energy	ExxonMobil (XOM)	Shell (RDSA.AS)	PetroChina (PTR)
Automotive	Tesla (TSLA)	Volkswagen (VOW3.DE)	Toyota (7203.T)
Food	Coca Cola (KO)	AB InBev (ABI.BR)	Kirin (2503.T)

II. Réalisation - Tâches

- Analyse de l’article source
- Compréhension du code source
- Correction et amélioration des fonctionnalités de bases
- Ajout de fonctionnalités d’évaluation
- Visualisation dynamique des résultats
- Application sur différents cours boursiers (Apple, Tesla, Amazon, etc.)
- Recherche des limites du code
- Ajout de la possibilité de *trade* en temps réel (Si le marché est ouvert)
- Jouer avec les paramètres de TDQN (et évaluation des résultats)

II. Réalisation - Difficultés et améliorations

Difficultés :

- Informations limités
- Stochastiques : séries temporelles
- Temps d'entraînement
- Diversités des cours boursières
- Surpasser les techniques de *trading* classiques
- Peu de confiance à l'IA

Améliorations :

- Essayer d'autres modèles : PPO, LSTM
- Comparer aux modèles classiques
- Analyser les résultats
- Caractériser les performances des cours boursiers selon les modèles
- Changer les paramètres du modèle
- Changer le nombre d'états / actions possibles (utiliser d'un % du capital)

III. Démonstration

Allons voir le code de l’algorithme de *trading* ainsi qu’une application !

Références bibliographiques

[1] *An application of deep reinforcement learning to algorithmic trading.* Thibaut Théate, Damien Ernst :
https://www.sciencedirect.com/science/article/pii/S0957417421000737

[2] Code source du *trading* l'algorithme de Thibaut Théate et Damien Ernst :
https://github.com/ThibautTheate/An-Application-of-Deep-Reinforcement-Learning-to-Algorithmic-Trading

[3] John Moody, Lizhong Wu, Yuanson Liao, Matthew Saffel. Performance functions and reinforcement learning for trading systems and portfolios.

https://onlinelibrary.wiley.com/doi/10.1002/%28SICI%291099-131X%281998090%2917%3A5/6%3C441%3A%3AAID-FOR707%3E3.0.CO%3B2-%23

[4] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, Qionghai Dai. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading.

https://ieeexplore.ieee.org/abstract/document/7407387/authors#authors

[5] Umesh Palai. RNN, LSTM, And GRU For Trading. https://blog.quantinsti.com/rnn-lstm-gru-trading/

[6] van Hasselt, H. P., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-Learning. CoRR, abs/1509.06461.

[7] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. CoRR, abs/1412.6980.

[8] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167.