# Deep Deterministic Portfolio Optimization

## Machine learning in finance: Theoretical foundations

Mats Bererd


# Objectifs

- Le *trading* d'action nécessite des prises de décisions dynamiques, *i.e.* prises en temps réel.
- Le *Deep Reinforcement Learning* (DRL) offre un cadre idéal pour développer des stratégies de *trading* en équilibrant l'arbitrage exploitation de connaissances / exploration de nouvelles stratégies.
- Ce type de modèle est difficile à expliquer et leur reproductibilité est souvent mise en doute
- But : comparer les performances de *Deep Deterministic Policy Gradient* (DDPG) à des solutions optimales connue d'environnement de *trading* conceptuellement simple.

# Optimisation dynamique pour l'allocation de portefeuille

Formellement, le problème d'optimisation dynamique s'écrit :

$$\max_{\{a_t \in \mathcal{A}\}} \mathbb{E}\left[\sum_{t=0}^{T-1} \text{rwd}_t(s_t, a_t, s_{t+1}, \xi_t)\right] \tag{1}$$

$$\text{s.t. } s_{t+1} = f_t(s_t, a_t, \eta_t)$$

avec $a_t \in \mathcal{A}$ l'action choisie dans l'espace des actions (on appelle $\{a_t\}$ la politique/le contrôle), $s_t \in \mathcal{S}$ l'état de l'environnement à la date $t$, $\xi_t$ et $\eta_t$ des bruits et $\text{rwd}_t$ est la fonction de gain perçu.

# Optimisation dynamique pour l'allocation de portefeuille

- On se restreint au cas unidimensionnel. L'investisseur alloue une portion $\pi_t \in \mathbb{R}$ dans un actif risqué.
- L'investisseur agit un changeant son allocation d'actif : $a_t = \pi_{t+1} - \pi_t$
- Le revenu $r_t$ se décompose en un terme prévisible $p_t$ et un bruit $\eta_t^{(r)}$
- $p_t$ est un processus AR(1) de paramètre $\rho$, de bruit $\eta_t^{(p)}$
- L'état $s_t = (\pi_t, p_t)$ contient toute l'information disponible à la date $t$

# Optimisation dynamique pour l'allocation de portefeuille

L'optimisation dynamique pour allocation de portefeuille s'écrit :

$$\max_{\{\pi_t\}} \mathbb{E}\left[\sum_{t=0}^{T-1} \pi_{t+1} r_{t+1} - \text{cost}(|a_t|) - \text{risk}(\pi_{t+1})\right] \tag{2}$$

$$\pi_{t+1} = \pi_t + a_t \tag{3}$$

$$p_{t+1} = \rho p_t + \eta_t^{(p)} \tag{4}$$

$$r_{t+1} = p_t + \eta_t^{(r)} \tag{5}$$

# Optimisation dynamique pour l'allocation de portefeuille

Deux sous-cas de figure sont alors étudiés :

- $r_{t+1} \mapsto p_t$ ($\eta_t^{(r)} = 0$ p.s.) : l'agent a une observation parfaite de l'espace des états en $t$. En effet, le gain perçu est fonction des paramètres observables $\pi_t$, $p_t$ et $a_t$.
- le revenu est perturbé par le bruit $\eta_t^{(r)}$

# Coût et risque quadratique

Sous ces hypothèses, le problème devient une Commande LQ (*Linear-Quadratic Regulator*, LQR) en temps discret et horizon fini :

$$\max_{\{\pi_t\}} \mathbb{E}\left[\sum_{t=0}^{T-1} \pi_{t+1} p_{t+1} - \Gamma a_t^2 - \lambda \pi_{t+1}^2\right] \tag{6}$$

La solution s'écrit :

$$\pi_{t+1}^* = (1-\omega)\pi_t + \omega\psi\pi_{t+1}^{(M)} \tag{7}$$

avec $\pi_{t+1}^{(M)} = \frac{p_t}{2\lambda}$, $\psi = \frac{\omega}{1-(1-\omega)\rho}$, $\omega = f_c\left(\sqrt{\frac{\lambda}{T}}\right)$ et

$f_c(x) = \frac{2}{1+\sqrt{1+\frac{4}{x}}} = \frac{x}{2}(\sqrt{x^2+4} - x)$

# Coût proportionnel et risque quadratique

$$\max_{\{\pi_t\}} \mathbb{E}\left[\sum_{t=0}^{T-1} \pi_{t+1} p_{t+1} - \Gamma|a_t| - \lambda \pi_{t+1}^2\right] \tag{8}$$

La solution s'écrit :

$$\pi^*_{t+1} = \begin{cases} u(\pi^{(M)}_{t+1}) & \text{si } \pi_t > u(\pi^{(M)}_{t+1}) \\ l(\pi^{(M)}_{t+1}) & \text{si } \pi_t < l(\pi^{(M)}_{t+1}) \\ \pi_t & \text{sinon} \end{cases} \tag{9}$$

Une bonne approximation est de prendre

$$u(\pi^{(M)}_{t+1}) = \pi^{(M)}_{t+1} + b \quad \text{et} \quad l(\pi^{(M)}_{t+1}) = \pi^{(M)}_{t+1} - b$$

# Coût proportionnel et risque *maxpos*

La contrainte de risque *maxpos* impose une limite maximum aux positions prises : $|\pi_t| \leq M$. Ce qui donne :

$$\max_{\{|\pi_t| \leq M\}} \mathbb{E}\left[\sum_{t=0}^{T-1} \pi_{t+1} p_{t+1} - \Gamma |a_t|\right] \tag{10}$$

La stratégie optimale est alors de trader le maximum autorisé quand le prédicteur $p_t$ dépasse un seuil :

$$\pi^*_{t+1} = \begin{cases} M & \text{si } \pi_t > q \\ -M & \text{si } \pi_t < -q \\ \pi_t & \text{sinon} \end{cases} \tag{11}$$

- DDPG est un algoritme acteur-critique
- La partie acteur permet d'estimer une politique déterministe $\phi_\Theta : \mathcal{S} \to \mathcal{A}$
- La partie critique donne une approximation de la fonction valeur $\mathcal{Q}$, satisfaisant l'équation de Bellman.
- L'espace des actions est considéré continu
- $\mathcal{Q}(s, a)$ est supposée différentiable en $a$.
- L'algorithme s'entraîne en *off-policy*.
- L'agent récupère de l'expérience sous forme de tuples $(s_t, a_t, \text{rwd}_t, s_{t+1})$ dans un *replay buffer*, puis échantillonne des nouvelles données d'entraînement pour mettre à jour la partie critique puis la partie acteur.

# Prioritized Experience Replay

- Cela permet de réduire le biais d'estimation
- Chaque « expérience » $(s_{t_j}, a_{t_j}, \text{rwd}_{t_j}, s_{t_{j+1}})$ est pondérée selon leur erreur TD :

$$\delta(s, a, \text{rwd}, s') = \text{rwd} + \gamma \max_{a'} \mathcal{Q}(s', a') - \mathcal{Q}(s, a)$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \ 1 \leq i \leq N, \ 0 \leq \alpha \leq 1$$

avec $N$ la taille du *replay buffer*, $p_i = |\delta_i| + \varepsilon$, $\varepsilon$ un petit nombre positif

- Pour l'actualisation de la partie critique, on corrige le biais d'échantillonage non-uniforme en utilisant :

$$\alpha_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^\beta, \ 0 \leq \beta \leq 1$$

# Exploartion de nouvelles stratégies

Afin d'explorer de nouvelles stratégies de *trading*, on applique une perturbation à la prédiction d'allocation : $a_t = a_t^{pred} + \eta_t^{(a)}$, étant donnée sa position actuelle $\pi_t$ et le revenu prévisible $p_t$ :

$$\begin{cases} a_t^{pred} = \phi_\Theta(p_t, \pi_t) \\ \eta_t^{(a)} = (1 - \rho^{expl})\eta_{t-1}^{(a)} + \sigma^{expl}\epsilon_t \\ \eta_0^{(a)} = 0 \end{cases}$$

$\eta_t^{(a)}$ est le bruit d'exploration, qui suit un processus AR(1) avec $\epsilon_t \sim_{iid} \mathcal{N}(0,1)$, $\sigma^{explo} > 0$.

- Pour améliorer la stabilité dans l'apprentissage, les paramètres sont actualisés sur des copies des modèle acteur et critique.
- Les réseaux finaux sont actualisés par moyennisation de Polyak :

$$\tilde{\omega} \leftarrow \tau_c\, \omega + (1 - \tau_c)\, \tilde{\omega},\ 0 < \tau_c < 1$$

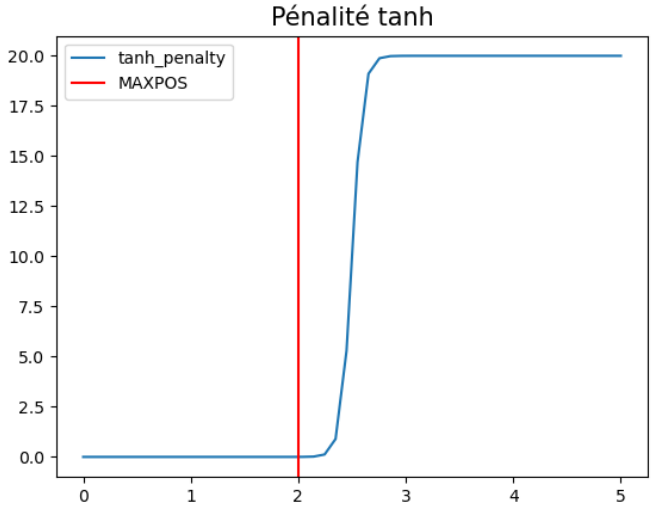
$$\tilde{\Theta} \leftarrow \tau_a\, \Theta + (1 - \tau_a)\, \tilde{\Theta},\ 0 < \tau_a < 1$$

avec $\omega, \tilde{\omega}, \Theta, \tilde{\Theta}$ les paramètres des réseaux copie critique, critique, copie acteur et acteur.

# Problème de différentiabilité de $\mathcal{Q}$

- les positions de l'acteur sont rognées pour respecter la contrainte
- un coût supplémentaire est ajouté à la fonction de gain :

$$\mathrm{rwd}(p,\pi,a) = p\pi - \Gamma|a| - \beta\{\tanh[\alpha(|\pi+a|+(1+\gamma)M] + 1\}$$



# Conclusions générales

- DDPG parvient à apprendre des stratégies de *trading* proche des solutions optimales en terme de fonction de gain et de *PnL*.
- Les performances au niveau *PnL* sont atteintes plus rapidement qu'au niveau fonction de gain.
- Ajouter du bruit à la fonction de gain est plus complexe à gérer : les résultats sont plus variables.
- Le troisième environnement est le plus difficile : plusieurs agents n'ont pas convergé.

# Résultats

| | Reward | | PnL | | Diff | |
|---|---|---|---|---|---|---|
| | no noise | noise | no noise | noise | no noise | noise |
| reference | 0.681 | | 1.298 | | 0 | |
| best | 0.677 | 0.671 | 1.291 | 1.674 | 0.081 | 0.128 |
| mean | 0.665 | 0.596 | 1.237 | 1.295 | 0.140 | 0.383 |
| worst | 0.655 | 0.415 | 1.170 | 1.119 | 0.218 | 0.781 |
| 75%-tile | 0.668 | 0.640 | 1.258 | 1.316 | 0.161 | 0.491 |
| 50%-tile | 0.666 | 0.624 | 1.239 | 1.276 | 0.132 | 0.349 |
| 25%-tile | 0.660 | 0.588 | 1.215 | 1.215 | 0.106 | 0.256 |

Table 1 – Résultats Coût et risque quadratique

# Résultats

| | Reward | | PnL | | Diff | |
|---|---|---|---|---|---|---|
| | no noise | noise | no noise | noise | no noise | noise |
| reference | 0.254 | | 0.492 | | 0 | |
| best | 0.248 | 0.225 | 0.518 | 0.562 | 0.063 | 0.131 |
| mean | 0.241 | 0.181 | 0.478 | 0.452 | 0.093 | 0.250 |
| worst | 0.234 | 0.135 | 0.442 | 0.364 | 0.126 | 0.441 |
| 75%-tile | 0.244 | 0.196 | 0.491 | 0.486 | 0.101 | 0.301 |
| 50%-tile | 0.239 | 0.188 | 0.482 | 0.455 | 0.090 | 0.244 |
| 25%-tile | 0.238 | 0.167 | 0.464 | 0.414 | 0.080 | 0.181 |

Table 2 – Résultats Coût proportionnel et risque quadratique

# Résultats

| | Reward | | PnL | | Diff | |
|---|---|---|---|---|---|---|
| | no noise | noise | no noise | noise | no noise | noise |
| reference | 0.901 | | 0.901 | | 0 | |
| best | 0.884 | 0.876 | 0.884 | 0.876 | 0.101 | 0.143 |
| mean | 0.856 | 0.842 | 0.856 | 0.842 | 0.198 | 0.246 |
| worst | 0.815 | 0.803 | 0.815 | 0.803 | 0.321 | 0.346 |
| 75%-tile | 0.849 | 0.849 | 0.849 | 0.849 | 0.148 | 0.210 |
| 50%-tile | 0.862 | - | 0.862 | - | 0.184 | - |
| 25%-tile | 0.826 | - | 0.826 | - | 0.239 | - |

Table 3 – Résultats Coût proportionnel et risque *maxpos*

# Limites et pistes d'amélioration

- DDPG suppose un espace d'actions continu.
- D'autres ajustements pourront sûrement améliorer la convergence et la précision de l'algorithme, *e.g.* changer la méthode d'exploration (ici $\epsilon$-greedy)
- DDPG est *model-free*, *i.e.* la fonction de gain est supposée non-connue au moment de l'entraînement. Ajouter ces informations de modèle peut aussi améliorer les performances.