# Soutenance Projet de Master

Romain Mencattini

Université de Genève

June 13, 2018

# Sommaire

# Notions de finance

Définition du FOREX[1]:

> *Forex (FX) is the market in which currencies are traded. The forex market is the largest, most liquid market in the world, with average traded values that can be trillions of dollars per day. It includes all of the currencies in the world.*

[1] Source : `https://www.investopedia.com/terms/f/forex.asp`

# Notions de finance

Propriétés du marché :

- Acteurs : les banques commerciales, les entreprises, les institutions financières non-bancaires et les ménages.
- Respecte la règle des trois unités.
- Outils possibles : *spot*,*futur* et *option*.

# Notions de *Machine learning*

Définition formelle du *Machine learning*[2]:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

[2]Source : "*Machine learning*" by Tom M. Mitchell

# Notions de *Machine learning*

Notions importantes :

- Données d'entraînement et de test.
- Apprentissage supervisé.
- Apprentissage non-supervisé.

# Motivations

Principales motivations :

- Peu de recherche sur le *machine learning* appliqué au FOREX.
- Prendre en main et améliorer une technique existante.

# Sommaire

# Introduction

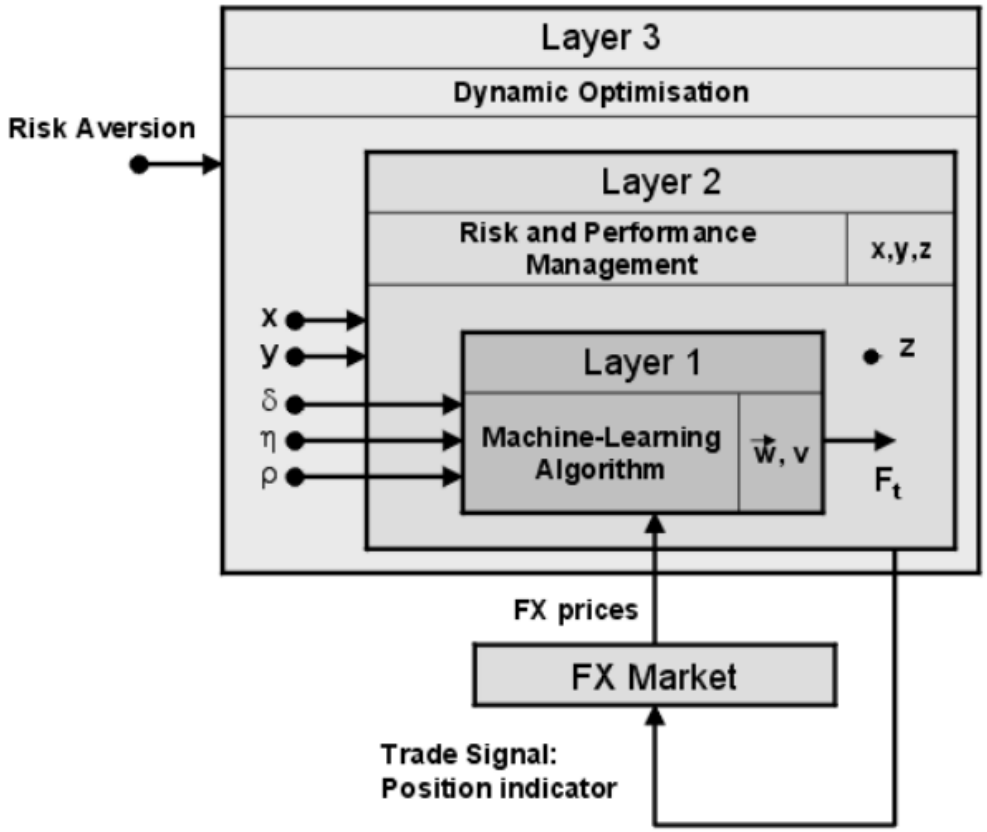


Figure: Image provenant de l'article "*An automated FX trading system using adaptive reinforcement learning*" de M.A.H. Dempster, V. Leemans

# Théorie *Layer 1*

Éléments de la *Layer 1* :

- Le calcul du signal :

$$F_t = sign(\sum_{i=0}^{M} w_{i,t} r_{t-i} + w_{M+1,t} F_{t-1} + v_t)$$

- L'optimisation s'effectue par descente du gradient. Le but est d'optimiser une estimation du *Sharpe ratio* :

$$\widehat{S}(t) := \frac{A_t}{B_t}$$

Où $A_t := A_{t-1} + \eta(R_t - A_{t-1})$ et $B_t := B_{t-1} + \eta(R_t^2 - B_{t-1})$

# Théorie *Layer 2*

La *Layer 2* est composée de:

- *stop loss*.
- seuillage.


Figure: Effet de la deuxième couche sur un profit cumulé de 3000 points.

# Théorie *Layer 3*

Éléments de la *Layer 3* :

- Les fonctions de coûts sont définies comme :

$$\Sigma := \frac{\sum_{i=0}^{N}(R_i)^2 I(R_i < 0)}{\sum_{i=0}^{N}(R_i)^2 I(R_i > 0)}$$

$$U(\overline{R}, \Sigma, \upsilon) := a \cdot (1 - \upsilon) \cdot \overline{R} - \upsilon \cdot \Sigma$$

avec $\overline{R} := \frac{P_N}{N}$, $\upsilon$ l'aversion au risque et $a$ une constante.

- L'optimisation est donnée par :

$$\max_{\delta,\eta,\rho,x,y} U(\overline{R}; \Sigma : \delta, \eta, \rho, x, y; \upsilon)$$

- Afin de rendre le calcul possible, cette formule est préférée :

$$\max_{\delta} \max_{\eta} \max_{\rho} \max_{x} \max_{y} U(\bar{R}; \Sigma : \delta, \eta, \rho, x, y; \nu)$$

# Implémentation

Principalement des choix sur ce qui n'était pas expliqué

# Problèmes

Les problèmes rencontrés sont :

1. Les poids ($w_i$) tendent vers l'infini.
2. L'apprentissage est globalement inefficace.
3. L'optimisation des méta-paramètres est bancale.
4. Le programme ralenti au fil de l'exécution.

# Solutions

Problème :

*Les poids ($w_i$) tendent vers l'infini.*

Solution :

*En changeant la descente du gradient par RMSProp :*

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

*avec*

$$\theta_{t+1} = \theta_t + \frac{\rho}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

*Cela permet d'éviter la divergence.*

# Solutions

Problème :

*L'apprentissage est globalement inefficace.*

Solution :

*Par l'utilisation du Sharpe ratio comme filtre, on diminue les périodes risquées et les pertes.*

# Solutions

Problème :

*L'optimisation des méta-paramètres est bancale.*

Solution :

*Fixer les valeurs initiales des méta-paramètres et effectuer une recherche aléatoire normale autour de ces dernières améliorent significativement l'optimisation.*

# Solutions

Problème :

*Le programme ralenti au fil de l'exécution.*

Solution :

*En évitant de stocker des éléments sur plusieurs itérations lorsque ce n'est pas nécessaire et en utilisant le parallélisme cela résous le problème.*

# Solutions

Comme solutions possibles :

1. Changer la descente du gradient par RMSProp :

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

avec

$$\theta_{t+1} = \theta_t + \frac{\rho}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

2. Utilisation du *Sharpe ratio* comme filtre.
3. Fixer les valeurs initiales des méta-paramètres et effectuer une recherche aléatoire normale autour de ces dernières.
4. Éviter de stocker des éléments sur plusieurs itérations si ce n'est pas nécessaire ainsi qu'une utilisation du parallélisme.

# Sommaire

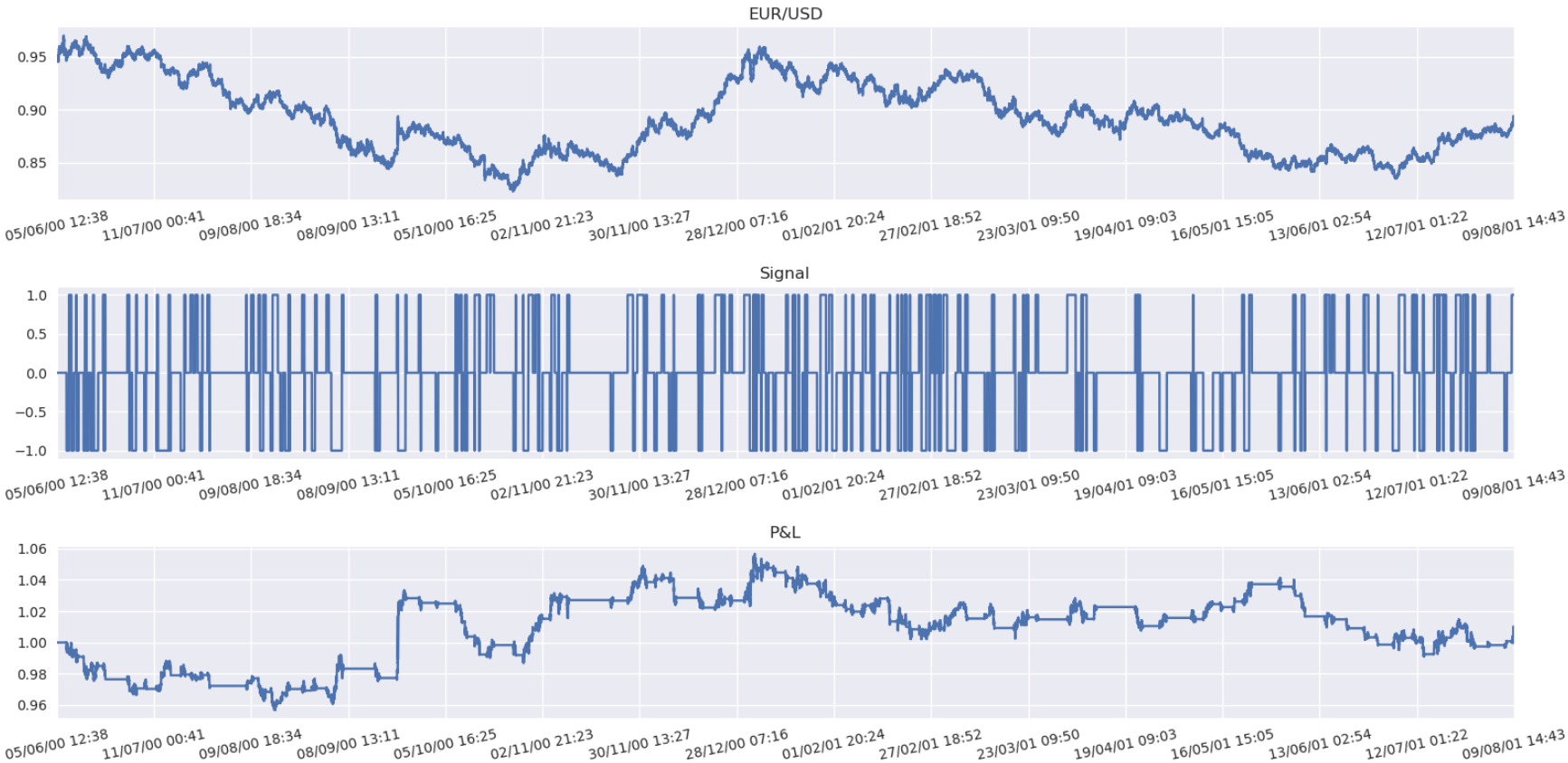


Figure: Rajouter légende

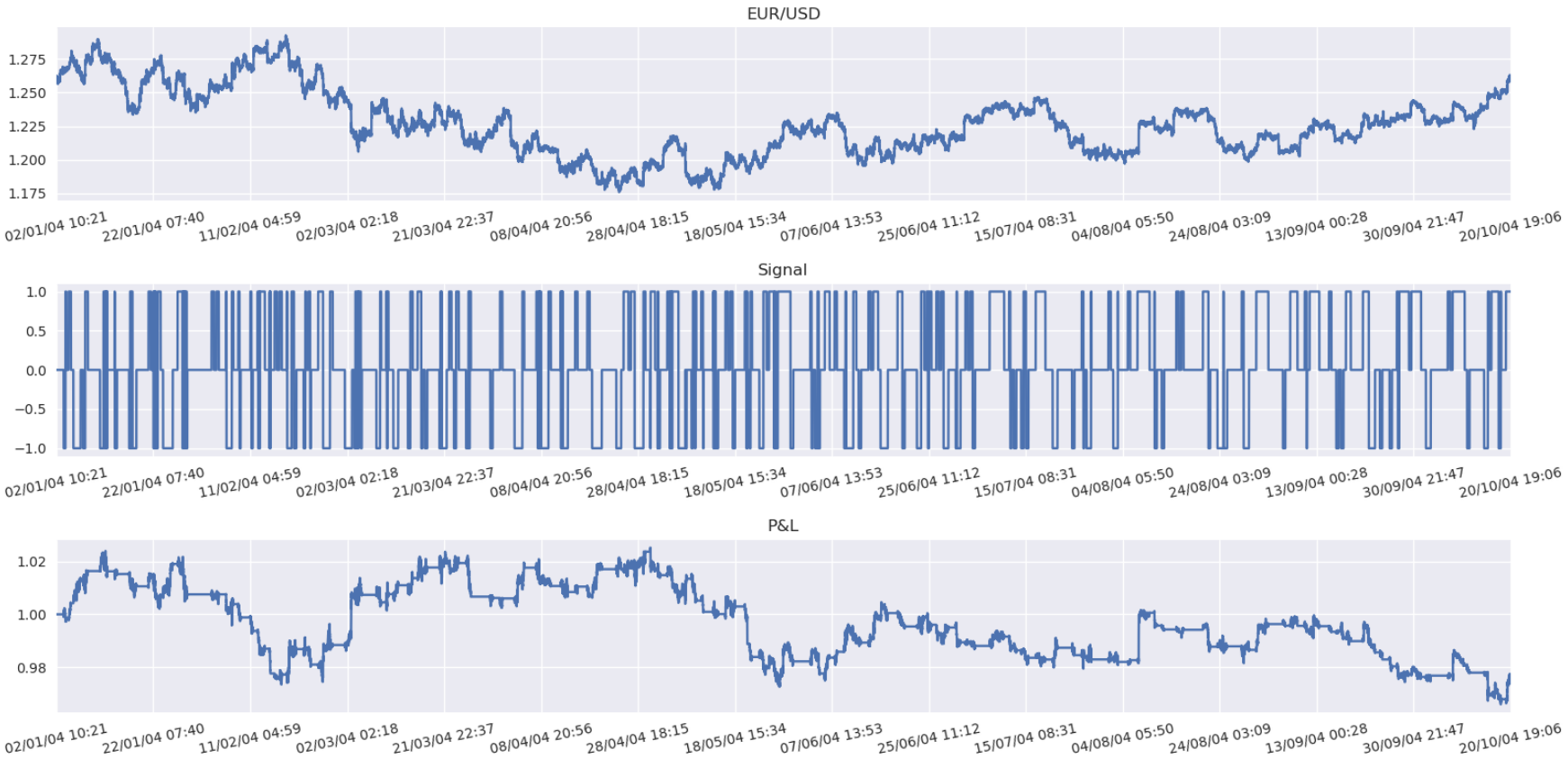


Figure: Rajouter légende


Figure: Rajouter légende

# Vitesse d'exécution

Petit benchmark

# Sommaire

# Pistes

Pistes envisageables :

- Ajouter des couches cachées au réseau de neurones.
- Améliorer l'optimisation des méta-paramètres.
- Implémenter une version alternative de la descente du gradient.
- Passer à des méthodes structurelles.

# Conclusion

En conclusion :

- Fonctionne localement mais pas globalement.
- Le projet est dans un état fonctionnel à partir d'un article incomplet.
- Cela fournit des pistes de recherches.