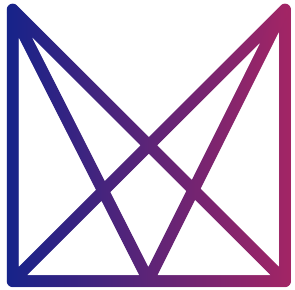

Modèles d'interpolations non-linéaires multidimensionnelles pour l'évaluation de produits exotiques

CORVISIER Jean-Christophe, KHELDOUNI Mohammed Amine

5 avril 2019

Plan de la présentation

Introduction

- Le *pricing* des instruments financiers tels que les options exotiques constitue une problématique majeure pour les institutions financières.
- Des solutions d'investissement et de couverture du risque utilisent des options exotiques basées sur un ou plusieurs ensembles de sous-jacents parmi les actions, les indices boursiers et d'autres produits dérivés.

Objectif : Mettre en place des algorithmes et modèles d'apprentissage (*Machine and Deep Learning*) pour prédire efficacement (en précision et en temps) le prix des options exotiques.

Données d'entrée

- La base de données d'entraînement est constituée d'un million d'exemples de produits exotiques paramétrés sur $d = 23$ *features* qui les représentent.
- A chaque exemple, un prix $y \in \mathbb{R}$ est donné pour l'ensemble d'entraînement.
- L'ensemble de validation et le score sur la plateforme est construit par la prédiction du prix de 400.000 produits dont la valeur réelle est inconnue.
- Les 23 variables proposées par Natixis sont représentatives du produit évalué et comprennent : les prix de trois sous-jacents ainsi que leur dérive, leur volatilité, leur maturité, leurs corrélations et d'autres données sur le produit (les prix des coupons, le Yeti coupon, la barrière Phénix, la barrière PDI ...).
- L'erreur est calculée comme étant l'erreur des moindres carrés entre le prix réel (*ground truth*) y_{test} et le vecteur des prédictions $\hat{y}_{test}$:

$$\mathcal{L} = \sum_{i=1}^{N_{test}} (y_{test} - \hat{y}_{test})^2$$

Visualisation des données et première approche

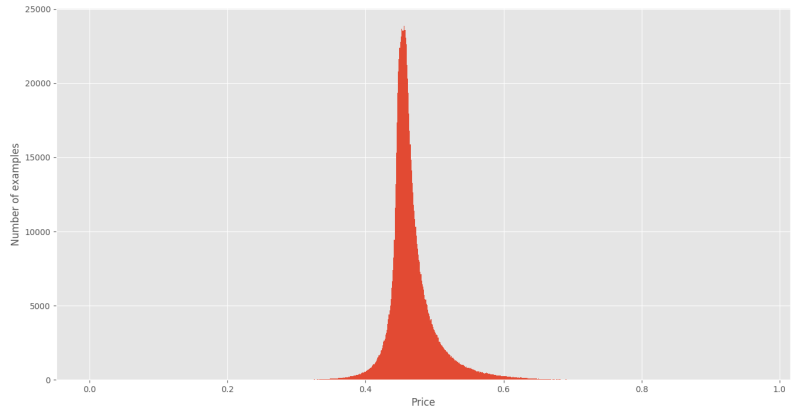

FIGURE – A gauche, un nuage de points modélisant les données d'entraînement agrégé en deux vecteurs propres (ACP). A droite, l'histogramme représente la distribution des prix de notre ensemble d'entraînement.

- Les exemples d'entraînement sont comparables et ne sont pas assimilables à des *clusters* remarquables.
- La distribution des prix normalisés des produits est gaussienne centrée en $\mu = 0.46$ avec une très faible variance.
- Les modèles de régression utilisés (*Ridge* et *Lasso*) n'ont pas donné de résultats concluants.

Réseaux de neurones profonds

- La majeure partie de notre travail d'implémentation et de modélisation a été orientée vers des approches de *Deep Learning* où nous avons élaboré des réseaux de neurones denses. (Implémentation en Python avec Keras)

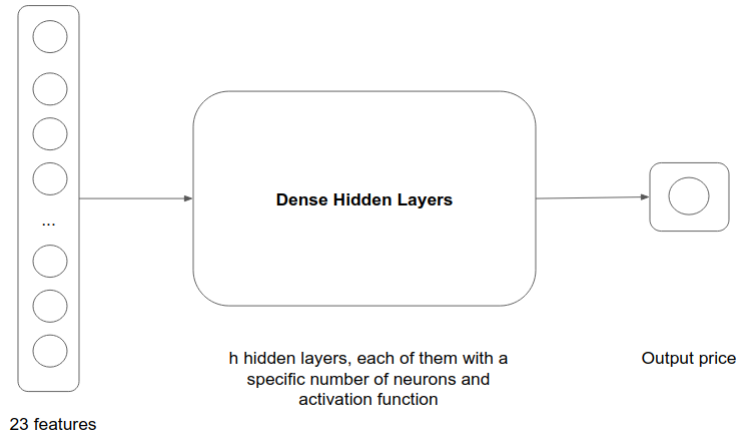

FIGURE – Schématisation du modèle de réseaux de neurones profonds

- Manque d'expressivité avec les méthodes de régression usuelles, d'où l'utilisation de fonctions non-linéaires

Calibration des paramètres

- Ainsi, on conçoit l'importance de calibrer les paramètres de nos modèles d'apprentissage à savoir :
  - Le nombre de couches h
  - Le nombre de neurones par couche n_h
  - Les fonctions d'activation
  - Les paramètres de l'optimiseur (Adam)
  - Le pas du gradient (*learning rate*)
  - Le nombre d'itérations (*epochs*)
  - La taille des batchs considérés (*batch size*)

Approche génétique pour la calibration de paramètres

- Optimisation de paramètres pour trouver de bonnes architectures
- Algorithme génétique sur une population de 10 réseaux
- Paramètres des réseaux de neurones modifiables : *learning rate*, *batch size*, nombre de couches, taille des couches, fonction d'activation.
- Nombre d'epochs fixé à 20, avec *early stopping*, et *Adam optimizer*
- Conservation des 3 meilleures architectures après chaque itération (10).
- Randomisation des paramètres en **mutation**, et entre les deux parents en ***cross-over***.

Informations obtenues

- *Learning Rate* compris entre 10^{-3} et 10^{-4}
- Choix d'un *batch size* faible (~ 256 exemples).
- Préférence pour les architectures profondes ($h \geq 8$) à grand nombre de neurones.
- Choix des fonctions d'activation non linéaires (ReLU et tanh).

Bulle de neurones et mutation de l'architecture

Procédure d'implémentation

- Construction d'un réseau de neurones initial $\mathcal{A}$.
- Mutation du réseau par l'ajout d'une bulle neuronale conservant la calibration de $\mathcal{A}$
- La bulle ajoutée est alors ré-entraînée pour améliorer la complexité du modèle et éviter les minima locaux.

Objectif de l'approche

Améliorer l'architecture en évitant les minima locaux

Résultat de notre implémentation

Nous n'avons réussi à reproduire les résultats démontrés par la *benchmark*.

Optimisation du pas d'apprentissage

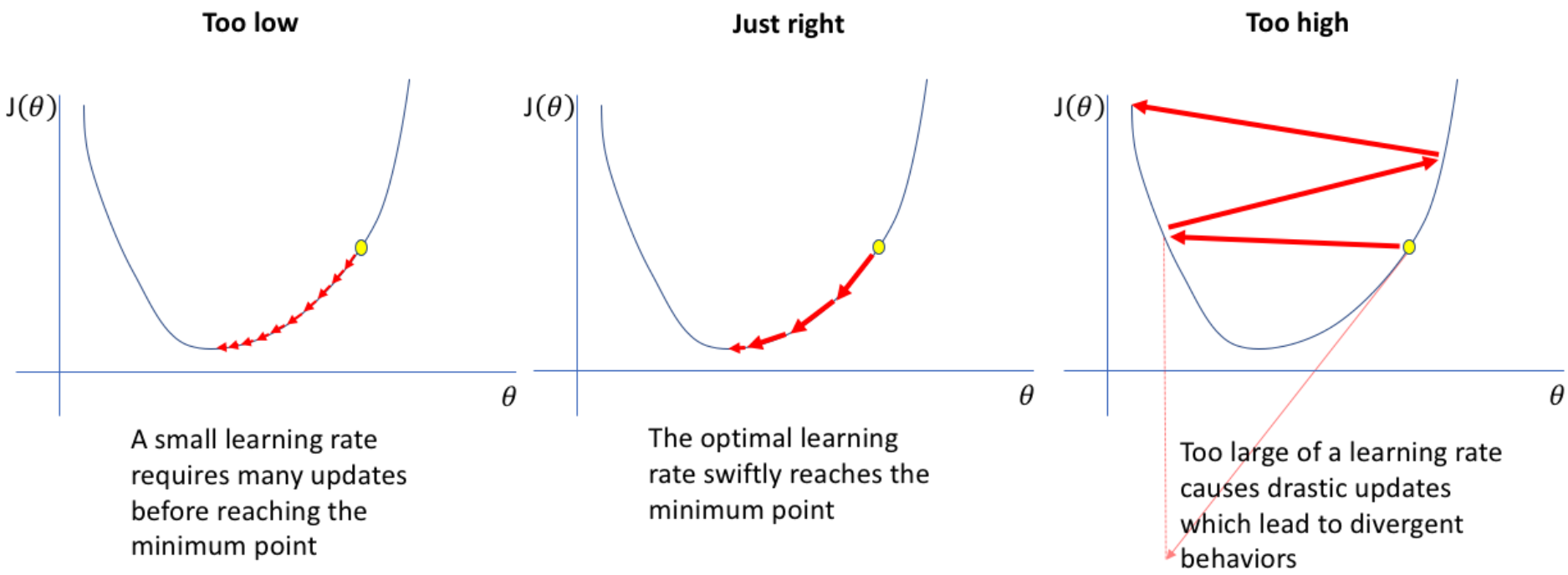

FIGURE – Propriétés basiques de la calibration du pas de gradient

Optimisation du pas d'apprentissage

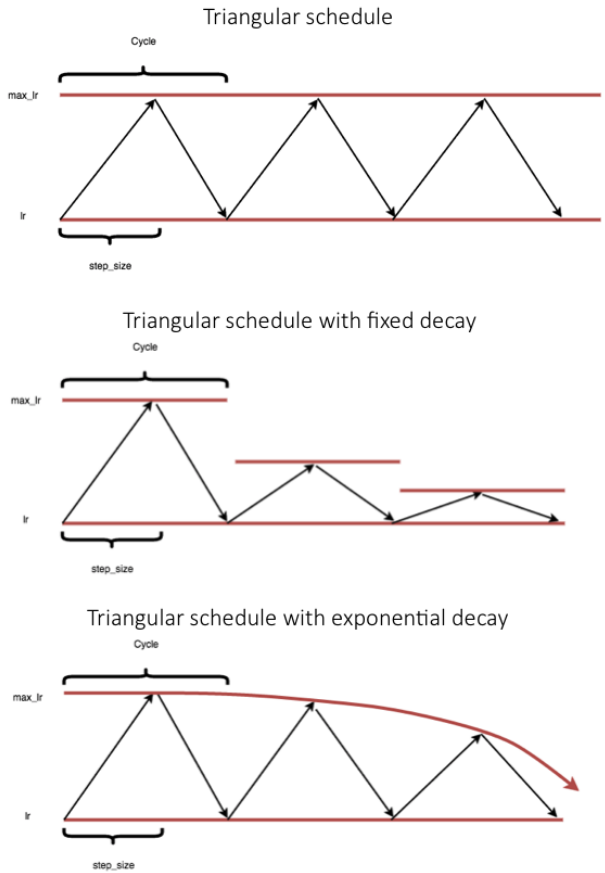

FIGURE – *Learning rate* triangulaire et décroissance exponentielle

Optimisation du pas d'apprentissage

Notre approche du problème des minima locaux

- Pour pallier à ce problème d'optimisation, nous avons décidé d'implémenter un pas d'apprentissage adaptatif en fonction des itérations, en augmentant parfois ce dernier afin de sortir des minima locaux.

- Deux types de pas d'apprentissage ont été implémentés
  - Un *learning rate* avec une décroissance exponentielle.
  - Un *learning rate* sinusoïdale avec décroissance exponentielle.

Visualisation des deux types de *learning rate*

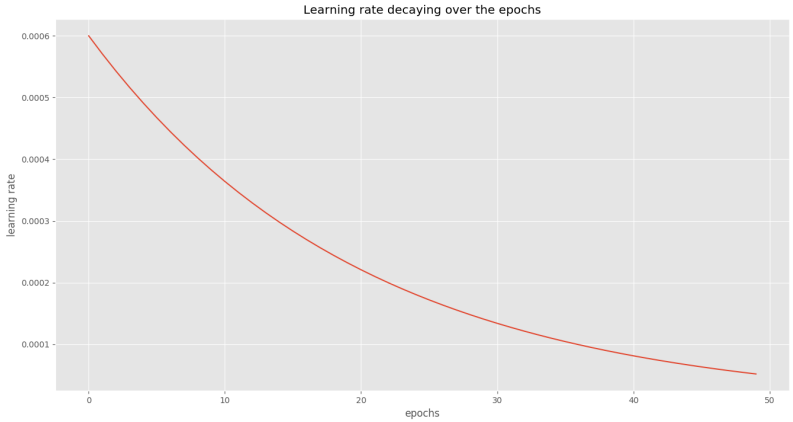

FIGURE – Décroissance exponentielle du pas du gradient

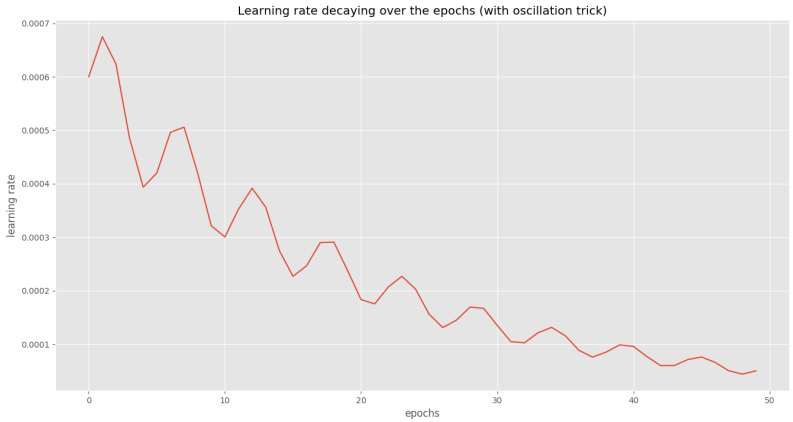

FIGURE – Décroissance exponentielle sinusoïdale du pas

Les modèles retenus

• $\mathcal{M}_1$: Réseaux de neurones dense avec 8 couches de neurones qui augmentent de 256 à 512 pour décroître à 1. Les activations considérées sont des tanh et des ReLU. Le learning rate initial a été fixé à 5×10^{-3} sur 30 epochs.

• $\mathcal{M}_2$: Même modèle que $\mathcal{M}_1$, mais utilisant un learning rate avec décroissance exponentielle sur 40 epochs d'entraînement.

• $\mathcal{M}_3$: Architecture similaire à $\mathcal{M}_2$ avec 10 couches augmentant jusqu'à 1024 neurones et une fonction d'activation linéaire sur la dernière couche.

• $\mathcal{M}_4$: Architecture profonde avec 10 couches avec 1024 neurones pour les premières couches avec une activation tanh et des ReLU sur les couches moins larges de neurones. Le *learning rate* initial a été fixé à 6×10^{-4} et subit la décroissance exponentielle avec de petites oscillations pour prévenir du risque de s'arrêter à un minimum local.

• $\mathcal{M}^*$: Modèle $\mathcal{M}_4$ ré-entraîné plusieurs fois sur quelques itérations.

Résultats obtenus

Modèle	$\mathcal{L}_{train}$	$\mathcal{L}_{val}$	Score publique	Score intermédiaire
$\mathcal{R}_1$	6.745×10^{-4}	6.70×10^{-4}	-	-
$\mathcal{R}_2$	6.72×10^{-4}	6.74×10^{-4}	-	-
$\mathcal{M}_1$	1.14×10^{-6}	1.76×10^{-6}	0.704	0.7183
$\mathcal{M}_2$	4.36×10^{-7}	8.57×10^{-7}	0.426	-
$\mathcal{M}_3$	8.76×10^{-7}	1.32×10^{-6}	0.526	-
$\mathcal{M}_4$	4.03×10^{-7}	5.416×10^{-7}	0.237	-
$\mathcal{M}^*$	3.66×10^{-7}	3.49×10^{-7}	0.2088	-
benchmark	-	-	0.1146	0.1145

TABLE – Table des erreurs des moindres carrés (moyenné par la taille de l'échantillon) pour les modèles sélectionnés

Public	Académique	Académique intermédiaire
5/20	2/3	2/3

TABLE – Classement de notre modèle sur la plateforme du Data Challenge

Convergence des modèles

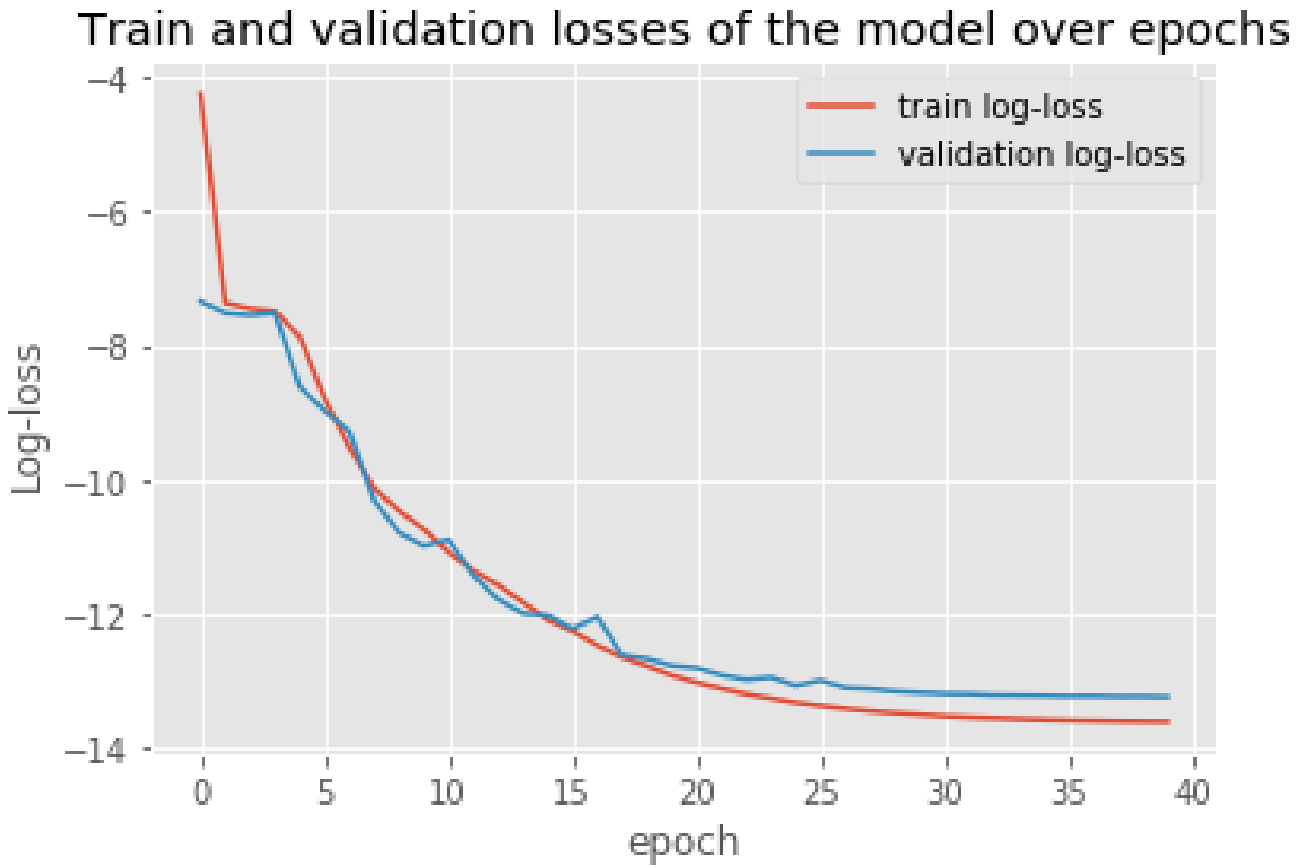

FIGURE – Tracé des erreurs en entraînement et en validation sur plusieurs itérations ($\mathcal{M}_2$)

Discussion des résultats

- Manque d'applicabilité des méthodes moins complexes que les réseaux de neurones profonds.
- Propriétés de calibration des paramètres révélées par l'algorithme génétique implémenté sur les réseaux de neurones.
- Nécessité d'un pas d'apprentissage variable en fonction des itérations pour pallier au problème des minima locaux.
- Implémentation d'architecture sans *overfitting* remarquable malgré la grande complexité des réseaux profonds utilisés : score public (0.704) comparable au résultat intermédiaire du 22 mars (0.7183).
- Axes d'amélioration possibles : modèles plus profonds, classifieur manifold tangentes ...

Conclusion

Au fil de ce projet, nous avons pu

- Travailler sur des données exhaustives et manipuler plusieurs variables financières pour expliquer et prédire le prix d'un produit exotique.
- Analyser les données et produire des algorithmes d'apprentissage basés sur des réseaux de neurones profonds pour capter les relations non-linéaires entre les variables explicatives.
- Calibrer les paramètres d'apprentissage de nos modèles et essayer diverses techniques d'optimisation pour améliorer la rapidité et la précision de nos prédictions.

Ce projet a été très intéressant et enrichissant et apporte une bonne application du *Deep Learning* et des modèles d'apprentissage dans les méthodes de *Pricing* en finance de marché !

Merci à vous !

References I

- Jeremy Jordan. Setting the learning rate of your neural network. (https://gist.github.com/jeremyjordan)
- Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. U.S. Naval Research Laboratory. (https://arxiv.org/pdf/1506.01186.pdf)
- Y. Bengio and M. Monperrus. Non-Local Manifold Tangent Learning.
- E. Zakkay. Advanced Keras - Accurately Resuming a Training Process (https://gist.github.com/eyalzk)
- A. Broström, R. Kristiansson. Exotic Derivatives and Deep Learning (KTH Royal Institute of Technology - School of Engineering Sciences)