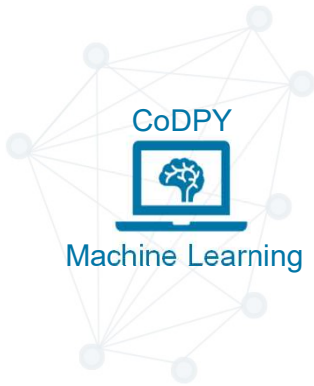


CoDPY est un framework alternatif aux plateformes traditionnelles de ML



Librairie de ML sous python / C++, open-source

Basée sur la théorie des noyaux reproduisants

Disposant d'une documentation solide écrite sous RMarkdown (~jupyter notebook)



Prédiction

Créer des modèles de prédiction performant en précision et temps de calcul



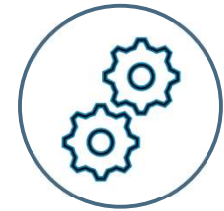
Boîte blanche

Expliquer chaque étape de la modélisation et créer une mesure d'erreur ex-ante sur chaque modèle



Segmentation

Segmenter des ensembles de données homogènes avec précision et stabilité



Analytics

Utiliser des fonctionnalités facilitant la traduction de problématiques d'ingénierie financière

CoDPY est une technologie mature qui a déjà fait ses preuves

20 publications

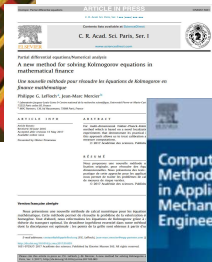
qui portent sur les fondements mathématiques de CoDePy et les algorithmes (Kernel Methods)

3 revues Alpha

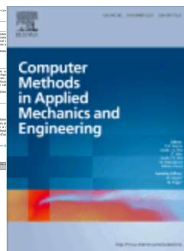


WILMOTT
MAGAZINE

COMPTE RENDU POUR
L'ACADEMIE DES
SCIENCES



COMPUTER METHODS
IN APPLIED MECHANICS
AND ENGINEERING



5 partenaires clés



QuantUniversity, LLC



8 années de R&D

Financées par MPG Partners et en étroite collaboration avec le Laboratoire Jacques Louis Lions du CNRS

6 use cases réalisés

- Couverture de bilan bancaire
- Optimisation du recouvrement
- Pricing de portefeuilles d'Autocall
- Reverse Stress Tests
- Pricing de prix futurs d'une option
- Fraudes de cartes bleus

3 use cases clients



CoDPY permet de faire des prédictions simplement avec d'excellentes performances

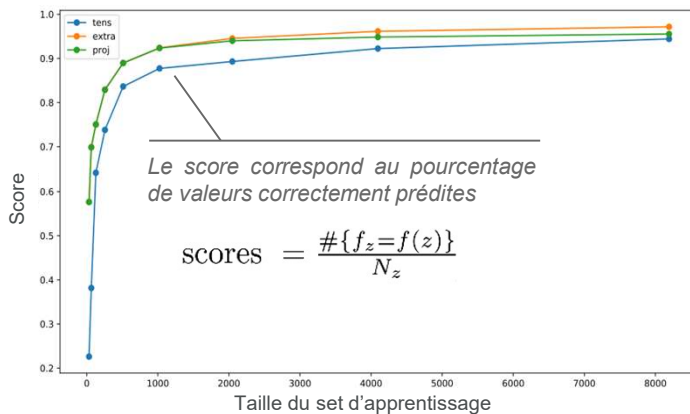


Prédiction

- Performance :** Cette technologie permet de construire des machines d'apprentissages qui surperforment systématiquement les grandes plateformes d'Intelligence Artificielle en terme de précision
- Explicabilité:** Le score obtenu peut être anticipé (par la mesure d'erreur, cf slide 14) et nos prédictions sont parfaitement explicables et auditable
- Nature des données :** La librairie CoDPY est particulièrement adaptée à des données parcimonieuses ;

Benchmark avec Google Tensorflow sur le test MNIST (KAGGLE)

Illustration 1 : Benchmark performance de prédiction entre CoDPY (projection et extrapolation) versus Google Tensorflow



- Google Tensorflow** (paramétrage standard)
- CoDPY Extrapolation** : (paramétrage standard)
Utilise toutes les données du set d'apprentissage. Précise mais coûteuse en temps de calcul.
- CoDPY Projection** : (paramétrage standard)
Utilise un sous ensemble du set d'apprentissage. Moins précise, mais très bonnes performances.



Simplicité du paramétrage sous CoDPY

```
codpy_param = {'set_kernel': 'gaussian'}
```



Complexité du paramétrage sous Tensorflow

```
import tensorflow as tf
tf_param = {'epochs': 10,
            'batch_size': 16,
            'validation_split': 0.1,
            'loss': tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            'optimizer': tf.keras.optimizers.Adam(0.001),
            'activation': ['relu', ''],
            'layers': [128, 10],
            'metrics': [tf.keras.metrics.SparseCategoricalAccuracy()]}
```

CoDPY permet de créer des modèles auditable parce qu'associés à une mesure d'erreur



Boîte blanche

● **Explicabilité** : cette mesure, quand calculée sur un set d'apprentissage et de test, permettent d'expliquer les résultats. **Essentiel ! Une méthode de cross validation ne suffit pas.**

● **Implémenté / testé sous python** : cette mesure est largement utilisée au sein de la librairie.

Notre approche de l'estimation d'erreur passe par la fonction de **mesure de discrédance** suivante, nous permettant d'évaluer **la worst error (une erreur ex-ante)** :

$$\|f(z) - f_z\| \leq d_k(x, y, z) \|f\|_{H_k}$$

Les écarts entre la fonction exacte et la fonction prédite sont bornés ...

... par la fonction de discrédance d_k , calculée sur les paramètres x , y et z qui représentent les différentes données utilisées (entraînement, weighted, tests) ...

... et la norme de la fonction dans un espace de Hilbert

CoDPY permet de reproduire les set d'apprentissage



Boîte blanche

- **Reproductibilité** : l'erreur de prédiction est nulle quand l'ensemble de test est le même que l'ensemble d'apprentissage.
- **Explicabilité** : la notion de reproductibilité permet d'interpréter et d'expliquer les résultats d'une prédiction comme une combinaison linéaire du set d'apprentissage.

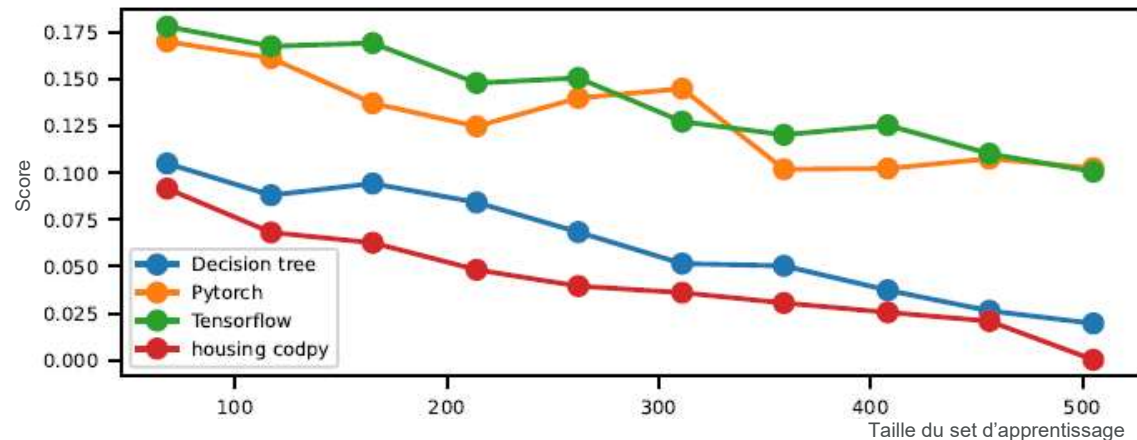
Benchmark avec les algorithmes **Decision Tree**, réseau de neurones de **Pytorch**, réseau de neurones **Google Tensorflow** sur le test Boston Housing Price (KAGGLE)

Le protocole de test consiste en une série de dix tests :

- L'apprentissage des modèles se fait par séquence croissante de 56 prix à 506 prix de maisons.
- La prédiction est faite sur le prix de l'ensemble des maisons (donc une partie de l'ensemble de test est comprise dans l'ensemble d'apprentissage).
- Reproductibilité : l'erreur sur le dernier point doit être nulle

Illustration 1 : Benchmark des scores de prédiction - Boston Housing Price

*Database: Boston Housing Prices (506 maisons – 13 caractéristiques)
Indicateurs de performance: RMSE %.*



CoDPY permet de faire de la segmentation de manière robuste



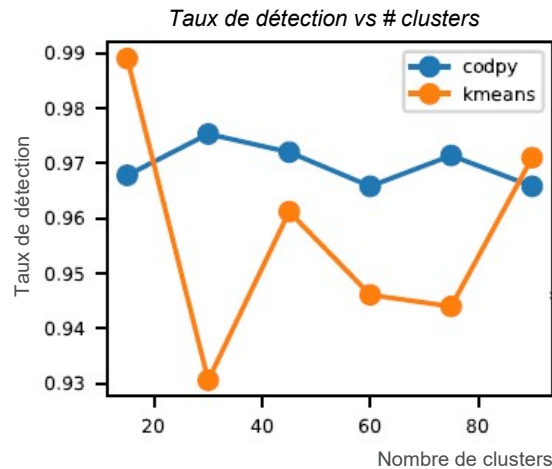
- Innovation** : les suites à faible discrédance sont un nouvel algorithme performant pour faire de la segmentation
- Stabilité** : cet algorithme est stable, la segmentation est déterministe et convergente

Benchmark avec l'algorithme **Kmeans de scikit-learn** sur un cas de fraude de cartes bleues (KAGGLE)

Illustration 1 : Benchmark de la stabilité et des scores de prédiction

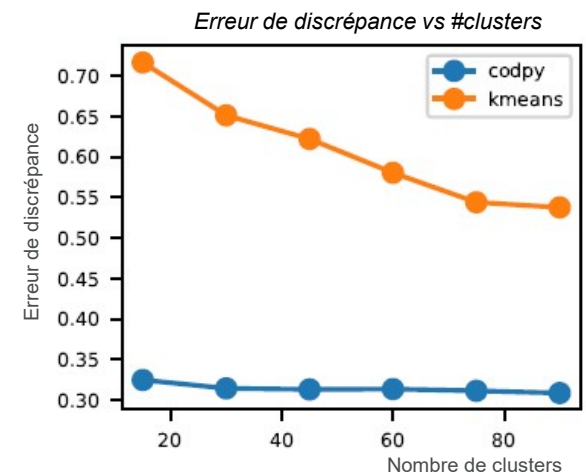
Database: Kaggle credit card fraud detection (284 000 lignes – 500 fraudes)
Indicateurs de performance: taux de détection / erreur de discrédance.
N = nombre de clusters

CoDPY est plus stable que KMeans



Nx

CoDPY est plus performant que KMeans



Nx

CoDPY permet d'accéder à des outils de simulations numériques pour le calcul de risques



- **Opérateurs différentiels** : Accès à tout opérateur différentiel (e.g. sensibilités / gammas et plus encore). Permet de résoudre une grande variété de problèmes décrits par les équations aux dérivées partielles.
- **Le calcul de sensibilité est très efficace**: performant, robuste et non intrusif, **meilleures performances que l'AAD**, pas de librairie tierce à installer.
- **Transport optimal**: Accès à des fonctionnalités de transport optimal. Pour le management des risques, seul ces outils permettent de calculer raisonnablement des **matrices de transitions de probabilité**, ou **générer des séries temporelles fiables** (🔍 Focus slide 18)

- Diffusion de sous-Jacents
- Valorisation
- Métriques de risques
- Stratégies de couverture

Exemples de problématiques	Fonctionnalités CoDPY	Avantages
<ul style="list-style-type: none"> - Réplication de séries temporelles (historique) Calibration des modèles paramétriques 	<ul style="list-style-type: none"> - Problème inverse : fonction de projection 	<ul style="list-style-type: none"> - Méthode généraliste - Méthode agnostique : pas d'hypothèse sur la distribution initiale - Simulation de trajectoires Monte-Carlo à partir de données historiques - Convergence rapide des résultats
<ul style="list-style-type: none"> - Modèles de diffusion stochastique (risque neutre): adéquation observations / modèles - Performance et précision au regard de la multitude de sources de risques 	<ul style="list-style-type: none"> - Fonction de sampling $z=S(x,N)$ - Calcul de suite de faibles discrétance pour les processus stochastiques 	
<ul style="list-style-type: none"> - Rétropropagation - Calcul de probabilités conditionnelles (CVA / risque de crédit) : Multitude de pricer / modèles - Calculs d'erreurs / intervalle de confiance 	<ul style="list-style-type: none"> - Calcul complet de la nappe des prix futurs - Fonction $fz x = Pi(x,z,f(z))$ 	<ul style="list-style-type: none"> - Calcul très efficace (pas de Monte-Carlo de Monte-Carlo) - Calcul de matrice de probabilité de transition - Estimation d'erreurs
<ul style="list-style-type: none"> - Calcul de sensibilités / dérivées - Evaluation de scénario 	<ul style="list-style-type: none"> - Fonction de dérivation : Nabla, Hessian, Divergence, autres 	<ul style="list-style-type: none"> - Calcul de sensibilité très rapide
<ul style="list-style-type: none"> - Evaluation de stratégies - Backtesting de stratégies 	<ul style="list-style-type: none"> - Optimisation sous contraintes 	<ul style="list-style-type: none"> - Evaluation de stratégies complexes (type américain / bermudéen ou hedging complexe)

Nos publications scientifiques se découpent en deux volets : une **partie théorique** qui porte sur les fondements mathématiques de nos méthodes innovantes et une **partie algorithmique** qui porte sur notre librairie d'algorithmes (Kernel Methods) :

Théorie

- "*A class of mesh-free algorithms for mathematical finance, machine learning and fluid dynamics* – Ce papier est l'essence même de notre approche ;
- "*The Transport-based Mesh-free Method (TMM) and its applications in finance: a review*" – publié dans Wilmott magazine, Ce papier est une présentation générale de notre approche ;
- "*Mesh-free error integration in arbitrary dimensions: A numerical study of discrepancy functions*" – publié dans Computer Methods in Applied Mechanics and Engineering. Ce papier se concentre sur l'erreur d'analyse pour la méthode Reproducing Kernel Hilbert Space (RKHS) ;
- "*A new method for solving Kolmogorov equations in mathematical finance*" - Comptes Rendus Mathématique, Volume 355, 6^{ème} édition, Juin 2017, Pages 680-686. Ce papier explique la stratégie des équations différentielles partielles pour les mathématiques financières et propose un certain nombre d'exemples numériques ;
- "*Revisiting the method of characteristics via a convex Hull algorithm*" – publié dans le Journal of Computational Physics, Octobre 2015 ([lien référence](#)) Ce papier traite de l'application de cette méthode pour les lois de conservation ;
- "*A high dimensional framework for financial instruments valuation*" – publié en 2013 ce papier est un premier essai pour décrire notre approche et la résolution d'équations aux dérivées partielles multidimensionnelle en mathématiques financières ;
- "*Optimally transported schemes*" – Papier portant sur le traitement pour le cas à une seule dimension ;

Algorithmes

- "*CoDPY-Tutorial*" – une première introduction de notre librairie, notamment sur le focus du Machine Learning ;
- "*CoDPY - Advanced Tutorial*" – une description technique de notre librairie CoDPY ;
- "*A kernel based reordering algorithm*" – description de l'algorithme central de reorganization de notre approche ;
- "*A kernel based polar factorization and the sampling algorithm with CoDPY*" –description de notre algorithme de calcul de factorisation polaire. Cet algorithme est illustré à travers un outil très pratique permettant de produire des échantillons IDD à partir de n'importe quelles distributions d'inputs ;
- "*A kernel-based algorithm to compute conditional expectations*" –un algorithm important pour les applications en Finance. Nous avons benchmarké l'implémentation de cet algorithme, en comparant notre framework à une approche Classique de Neural Network ;
- "*Hedging Strategies for Net Interest Income and Economic Values of Equity*" – une description d'un prototype utilisant CoDPY, ayant pour but la construction de stratégies sophistiquées sur des sujets d'ALM ;
- "*Kernel methods for stress test and reverse stress test*" – description de notre approche Support Vector Machine avancée aux Stress Tests et Reverse Stress Tests ;



Focus sur la fonction de réplcation de séries temporelles (sampling)



Analytics

Algorithmes de factorisation polaire : cette collection d'outils provenant du transport optimal nous donne accès à des algorithmes inédits en Machine Learning, tel que la factorisation polaire. La fonction de sampling que nous proposons est une application directe de cette factorisation ;

Fonction de sampling : un outil nous permettant de re-sampler n'importe quelle distribution. Très pratique pour générer des données de test ! Sa fonction est de générer des échantillons partageant les mêmes propriétés statistiques que la distribution originale. Essentiel pour la **simulation de trajectoires Monte-Carlo** à partir de données historiques (séries temporelles).

Notre fonction de sampling :

En entrée (inputs)

- des échantillons iid d'une variable aléatoire

En sortie (output) :

une distribution partageant des propriétés statistiques proches de x

Illustration (en 3D sur deux axes) d'une distribution multidimensionnelle bimodale, avec la distribution originale (en bleu) versus la distribution générée (en rouge)

