









# CodPy : a Python library for machine learning, mathematical finance, and statistics

Philippe G. LeFloch<sup>1</sup>, Jean-Marc Mercier, and Shohruh Miryusupov<sup>2</sup>

2021-06-04

-  **1 Introduction**
-  2 Brief overview of methods of machine learning
-  3 Kernel and differentiation techniques for machine learning
-  4 Kernel methods for optimal transportation
-  5 Supervised Machine Learning applications
-  6 Unsupervised Machine Learning Applications
-  7 Optimal transportations applications
-  8 Partial Differential Equations Applications



**Activities** : MPG Partners is a mid-sized French consulting firm specialized in risk management : ~ 50 employees / 22 clients



### Small size R&D team

- Jean-Marc Mercier [jean-marc.mercier@mpg-partners.com](mailto:jean-marc.mercier@mpg-partners.com)
- Shohruh Miryusupov [shohruh.miryusupov@mpg-partners.com](mailto:shohruh.miryusupov@mpg-partners.com)



Philippe LeFloch

[contact@philippelefloch.org](mailto:contact@philippelefloch.org)



# CodPy is a python library

**kernel-based** (RKHS - Reproducing Kernel Hilbert Space).

C++ core / python or graph database (xquery / xml based) interface

- **Compete with**

Tensorflow, Pytorch, Theano, scikit-learn, xgboost, ADA boost, k-trees, Random Forest,...)

(Artificial Intelligence, Neural Networks, Deep Learning, Support Vector Machine, k-trees, etc...)

- **Users** : 2014 - today : **internal library**. Tomorrow : **public access ? (pip install CodPy) ?**

- **Our user manuals are publicly available** (see last slide - bibliography)

- 1 Introduction
- 2 Brief overview of methods of machine learning
- 3 Kernel and differentiation techniques for machine learning
- 4 Kernel methods for optimal transportation
- 5 Supervised Machine Learning applications
  - 5.1 the MNIST problem - supervised learning
  - 5.2 Reconstruction from sub-sampled signals.
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

# supervised learning methods

1. Supervised learning

- 1.1. Linear Models
  - 1.1.1. Linear Regression
  - 1.1.2. Logistic Regression
  - 1.1.3. Support Vector Machines
  - 1.1.4. Linear Support Vector Machines
  - 1.1.5. Linear Support Vector Machines (SVM)
  - 1.1.6. Linear Support Vector Machines (SVM)
  - 1.1.7. Linear Support Vector Machines (SVM)
  - 1.1.8. Linear Support Vector Machines (SVM)
  - 1.1.9. Linear Support Vector Machines (SVM)
  - 1.1.10. Linear Support Vector Machines (SVM)
  - 1.1.11. Linear Support Vector Machines (SVM)
  - 1.1.12. Linear Support Vector Machines (SVM)
  - 1.1.13. Linear Support Vector Machines (SVM)
  - 1.1.14. Linear Support Vector Machines (SVM)
  - 1.1.15. Linear Support Vector Machines (SVM)
  - 1.1.16. Linear Support Vector Machines (SVM)
  - 1.1.17. Linear Support Vector Machines (SVM)
  - 1.1.18. Linear Support Vector Machines (SVM)
  - 1.1.19. Linear Support Vector Machines (SVM)
  - 1.1.20. Linear Support Vector Machines (SVM)
  - 1.1.21. Linear Support Vector Machines (SVM)
  - 1.1.22. Linear Support Vector Machines (SVM)
  - 1.1.23. Linear Support Vector Machines (SVM)
  - 1.1.24. Linear Support Vector Machines (SVM)
  - 1.1.25. Linear Support Vector Machines (SVM)
  - 1.1.26. Linear Support Vector Machines (SVM)
  - 1.1.27. Linear Support Vector Machines (SVM)
  - 1.1.28. Linear Support Vector Machines (SVM)
  - 1.1.29. Linear Support Vector Machines (SVM)
  - 1.1.30. Linear Support Vector Machines (SVM)
  - 1.1.31. Linear Support Vector Machines (SVM)
  - 1.1.32. Linear Support Vector Machines (SVM)
  - 1.1.33. Linear Support Vector Machines (SVM)
  - 1.1.34. Linear Support Vector Machines (SVM)
  - 1.1.35. Linear Support Vector Machines (SVM)
  - 1.1.36. Linear Support Vector Machines (SVM)
  - 1.1.37. Linear Support Vector Machines (SVM)
  - 1.1.38. Linear Support Vector Machines (SVM)
  - 1.1.39. Linear Support Vector Machines (SVM)
  - 1.1.40. Linear Support Vector Machines (SVM)
  - 1.1.41. Linear Support Vector Machines (SVM)
  - 1.1.42. Linear Support Vector Machines (SVM)
  - 1.1.43. Linear Support Vector Machines (SVM)
  - 1.1.44. Linear Support Vector Machines (SVM)
  - 1.1.45. Linear Support Vector Machines (SVM)
  - 1.1.46. Linear Support Vector Machines (SVM)
  - 1.1.47. Linear Support Vector Machines (SVM)
  - 1.1.48. Linear Support Vector Machines (SVM)
  - 1.1.49. Linear Support Vector Machines (SVM)
  - 1.1.50. Linear Support Vector Machines (SVM)
- 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- 1.4. Support Vector Machines
- 1.5. Stochastic Gradient Descent
- 1.6. Nearest Neighbors
- 1.7. Gaussian Processes
- 1.8. Cross decomposition
- 1.9. Naive Bayes
- 1.10. Decision Trees
- 1.11. Ensemble methods
- 1.12. Multiclass and multitarget algorithms
- 1.13. Feature selection
- 1.14. Semi-supervised learning
- 1.15. Isotonic regression
- 1.16. Probability calibration
- 1.17. Neural network models (supervised)

# unsupervised learning methods

2. Unsupervised learning

- 2.1. Gaussian mixture models
  - 2.1.1. Gaussian Mixture
  - 2.1.2. Variational Bayesian Gaussian Mixture
- 2.2. Manifold learning
  - 2.2.1. t-SNE
  - 2.2.2. UMAP
  - 2.2.3. Locally Linear Embedding
  - 2.2.4. Modified Locally Linear Embedding
  - 2.2.5. Process Embedding
  - 2.2.6. Spectral Embedding
  - 2.2.7. Local Tangent Space Alignment
  - 2.2.8. Multi-dimensional Scaling (MDS)
  - 2.2.9. t-Stochastic Neighbor Embedding (t-SNE)
  - 2.2.10. Open-Gradient-Descent
- 2.3. Clustering
  - 2.3.1. Overview of clustering methods
  - 2.3.2. K-means
  - 2.3.3. Affinity Propagation
  - 2.3.4. Mean Shift
  - 2.3.5. Spectral Clustering
  - 2.3.6. Hierarchical Clustering
  - 2.3.7. DBSCAN
  - 2.3.8. OPTICS
  - 2.3.9. BRISQ
  - 2.3.10. Clustering performance evaluation
- 2.4. Biclustering
  - 2.4.1. Spectral Co-Clustering
  - 2.4.2. Spectral Biclustering
  - 2.4.3. Biclustering evaluation
- 2.5. Decomposing signals in components (matrix factorization problems)
  - 2.5.1. Principal component analysis (PCA)
  - 2.5.2. Truncated singular value decomposition and latent semantic analysis
  - 2.5.3. Dictionary Learning
  - 2.5.4. Factor Analysis
  - 2.5.5. Independent component analysis (ICA)
  - 2.5.6. Non-negative matrix factorization (NMF or NNMF)
  - 2.5.7. Latent Dirichlet Allocation (LDA)
- 2.6. Covariance estimation
  - 2.6.1. Empirical covariance
  - 2.6.2. Shrinkage Covariance
  - 2.6.3. Sparse matrix covariance
  - 2.6.4. Robust Covariance Estimation
- 2.7. Novelty and Outlier Detection
  - 2.7.1. Overview of outlier detection methods
  - 2.7.2. Isolation Forest
  - 2.7.3. One-Class SVM
  - 2.7.4. Novelty detector with Local Outlier Factor
- 2.8. Density Estimation
  - 2.8.1. Density Estimation: Histograms
  - 2.8.2. Kernel Density Estimation
- 2.9. Neural network models (unsupervised)
  - 2.9.1. Restricted Boltzmann machines

# Performance indicators

Classification metrics

Regression metrics

Multitask ranking metrics

Clustering metrics

Biclustering metrics

Regression metrics

# Libraries

## List of IA libraries on PiPy

3,815 projects with the selected classifier

TOPIC :: SCIENTIFIC/ENGINEERING :: ARTIFICIAL INTELLIGENCE



- > 1 Introduction
- > 2 Brief overview of methods of machine learning
  - 2.1 A framework for machine learning
  - 2.2 Exploratory data analysis
  - 2.3 Performance indicators for machine learning
  - 2.4 Benchmark methodology for supervised learning: simple examples
  - 2.5 Benchmark methodology for unsupervised learning: simple examples
- > 3 Kernel and differentiation techniques for machine learning
- > 4 Kernel methods for optimal transportation
- > 5 Supervised Machine Learning applications
- > 6 Unsupervised Machine Learning Applications
- > 7 Optimal transportations applications
- > 8 Partial Differential Equations Applications

+1:  
kernel Projection  
(multi-output / labelled-unlabelled supervised learning machine)



+1:  
Sharp discrepancy sequences  
(clustering method)

+2:  
Discrepancy error Functional norms

+1:  
CodPy

# supervised learning methods : A methodology to benchmark them all

## Prediction function $f_z = P(x, y, z, f(x))$

- $x \in \mathbb{R}^{N_x \times D}$  training set.
- $y \in \mathbb{R}^{N_y \times D}$  weight set.
- $z \in \mathbb{R}^{N_z \times D}$  test set.
- $f(x) \in \mathbb{R}^{N_x \times D_f}$  multi-valued

- 1 Introduction
- 2 Brief overview of methods of machine learning
  - 2.1 A framework for machine learning
  - 2.2 Exploratory data analysis
  - 2.3 Performance indicators for machine learning
  - 2.4 Benchmark methodology for supervised learning: simple examples
  - 2.5 Benchmark methodology for unsupervised learning: simple examples
- 3 Kernel and differentiation techniques for machine learning
- 4 Kernel methods for optimal transportation
- 5 Supervised Machine Learning applications
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

**STEP I: pick up some libraries, and start setting their inner parameters**

### Neural Networks settings

```
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()]}
```

### AdaBoost settings

```
ada_param = {'tree_no': 50,
'base_model':'tree_model',
'learning_rate': 1,
'algorithm':'SAMME.R'}
```

### Random Forest settings

```
RF_param = {'max_depth': 5,
'n_estimators': 5}
```

### Xgboost settings

```
xgb_param = {'max_depth': 5,
'n_estimators': 10,
'num_class' : 10,
'objective':'multi:softmax',
'eta': 0.3,
'min_child_weight' : 1,
'num_boost_round': 100,
'eval_metric': 'mlogloss',
}
```

### CodPy settings

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

## supervised learning methods : the MNIST database (Y. Le Cun 1998)

- **II-a pick-up a data set.** MNIST (pattern recognition problem)
- **II-b Pick-up performance indicators :** accuracy scores, discrepancy errors
- **II-c Pick-up a list of tests :** variable training set size (**Nx**)

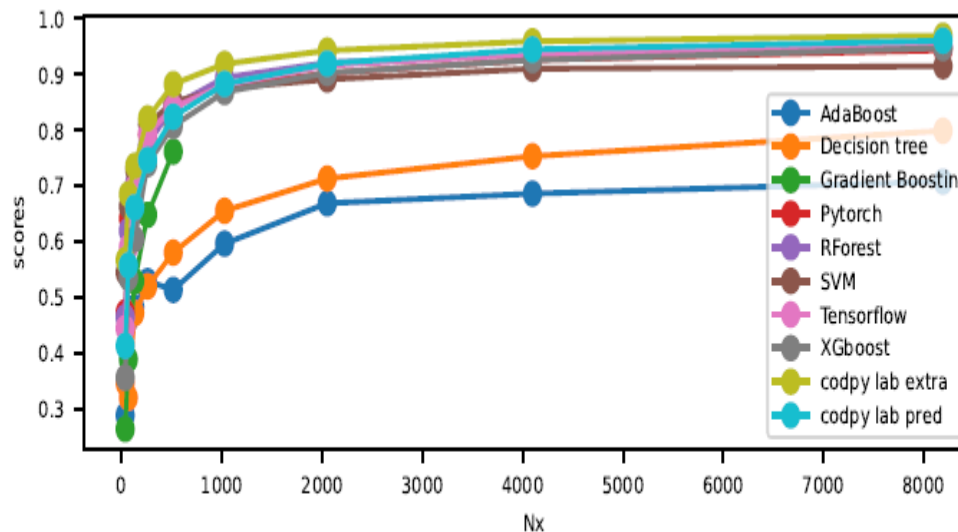


Figure 5.2: Benchmark scores

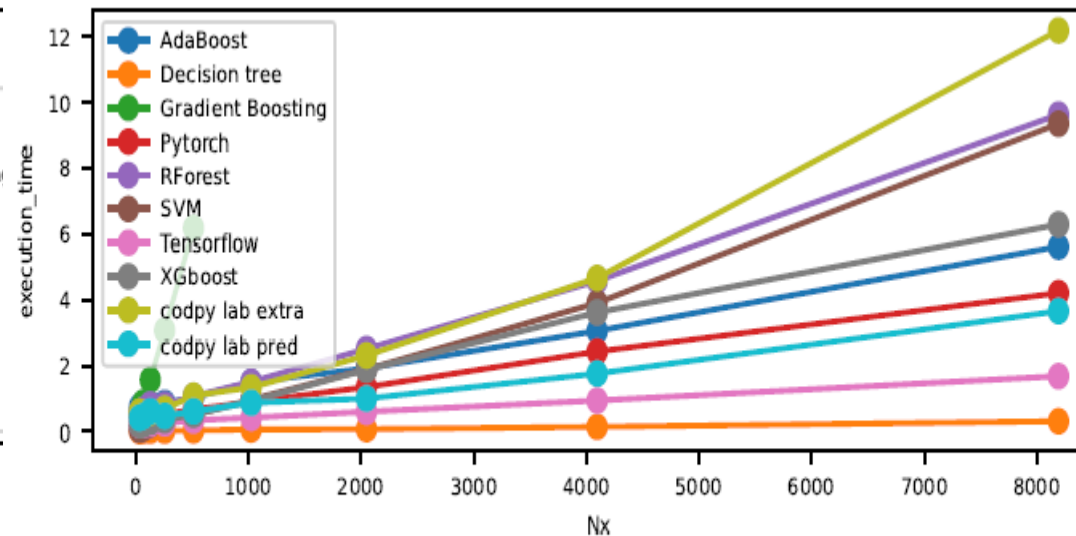


Figure 5.4: Benchmarks execution time

- > > 1 Introduction
- > > 2 Brief overview of methods of machine learning
- > > 3 Kernel and differentiation techniques for machine learning
- > > 4 Kernel methods for optimal transportation
- > > 5 Supervised Machine Learning applications
  - > > > 5.1 the MNIST problem - supervised learning
  - > > > 5.2 Reconstruction from sub-sampled signals.
- > > 6 Unsupervised Machine Learning Applications
- > > 7 Optimal transportations applications
- > > 8 Partial Differential Equations Applications

# What differentiate kernel methods from other machine learning technics ?

## Error estimates

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

- $d_k(x, y, z)$  discrepancy error.
- $\|f\|_{H_k}$  function norm.

**Both python functions**

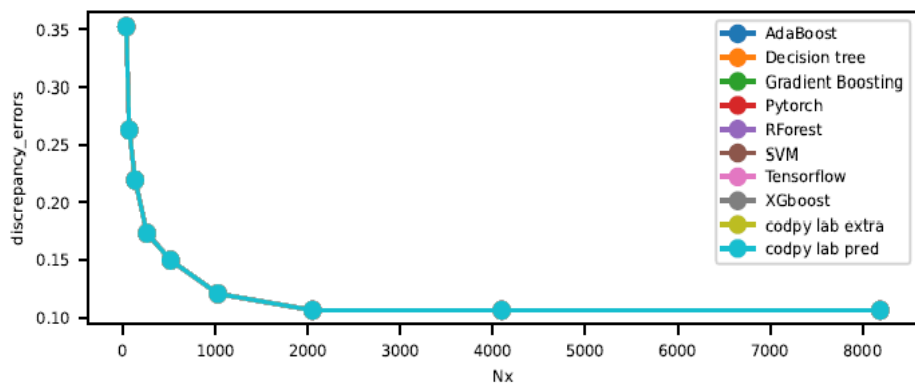


Figure 5.3: Discrepancy errors

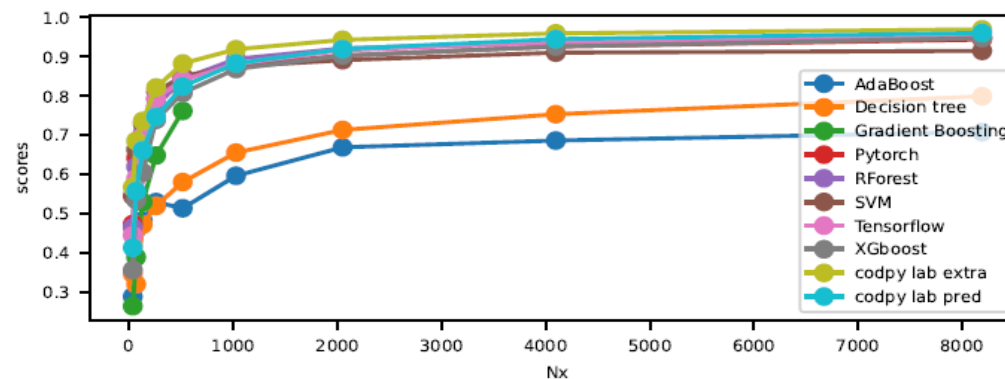


Figure 5.2: Benchmark scores

Kernel-based scores for MNIST can be **predicted** by discrepancy errors.  
**Codpy score  $\approx 1 - d(x, y, z)$**

- > 1 Introduction
- > 2 Brief overview of methods of machine learning
- > 3 Kernel and differentiation techniques for machine learning
  - 3.1 Aim of this chapter
  - 3.2 Fundamental notions for supervised learning
  - 3.3 Kernel engineering
  - 3.4 Operations with kernels
  - 3.5 Kernel operators
  - 3.6 A kernel-based clustering algorithm
- > 4 Kernel methods for optimal transportation
- > 5 Supervised Machine Learning applications
- > 6 Unsupervised Machine Learning Applications
- > 7 Optimal transportations applications
- > 8 Partial Differential Equations Applications

What differentiate kernel methods from other machine learning ?

## Error estimates Part II

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

- $d_k(x, y, z)$  discrepancy error.
- $\|f\|_{H_k}$  function norm.

**Each kernel-based learning machine is shipped with quantified, worst-error analysis tools for supervised / unsupervised learning**

**Consequence 1 : kernel-based methods are CONVERGENT**

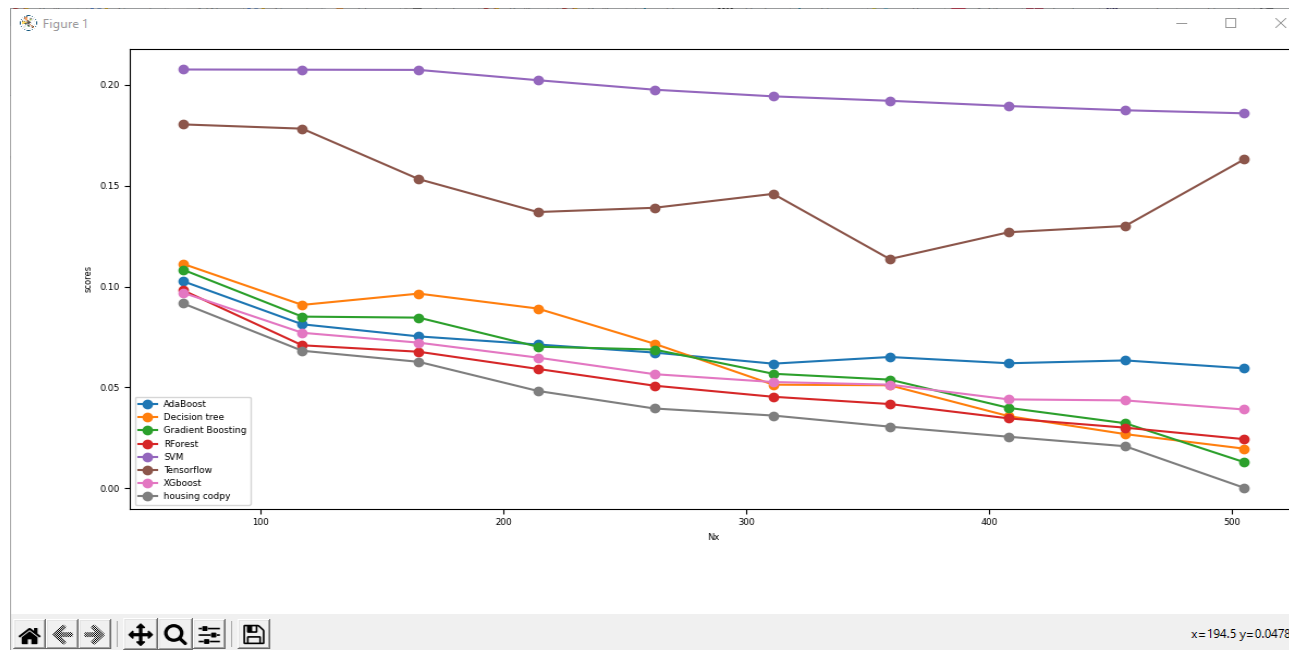
**Consequence 2 : kernel-based methods are EXPLAINABLE, hence AUDITABLE**



## Another benchmark: the venerable **BOSTON** Housing price (1970) (market price prediction)

- **II-a pick-up a data set.** BOSTON Housing prices
- **II-b Pick-up performance indicators :** Round Mean Square error %
- **II-c Pick-up a list of tests :** variable training set size (Nx)

Test set =all Boston database (506 entries)  
**the training set is part of the test set**



### Note:

**score of the kernel method is zero on the last point (perfect score) :**

**kernel methods can replicate exactly the training set.**

# Reconstruction from sub-sampled signals.

medical imagery, oceanography, petroleum, defense, astrophysics ...

## Application example: reconstruction from sub-sampled signals for SPECT machine

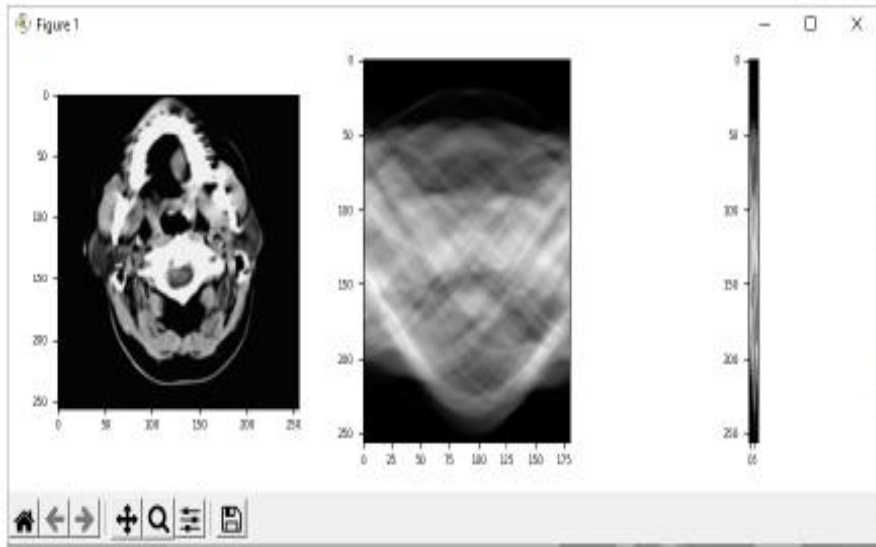


Figure 5.5: high resolution sinogram (middle), low resolution (right), reconstructed image (left)

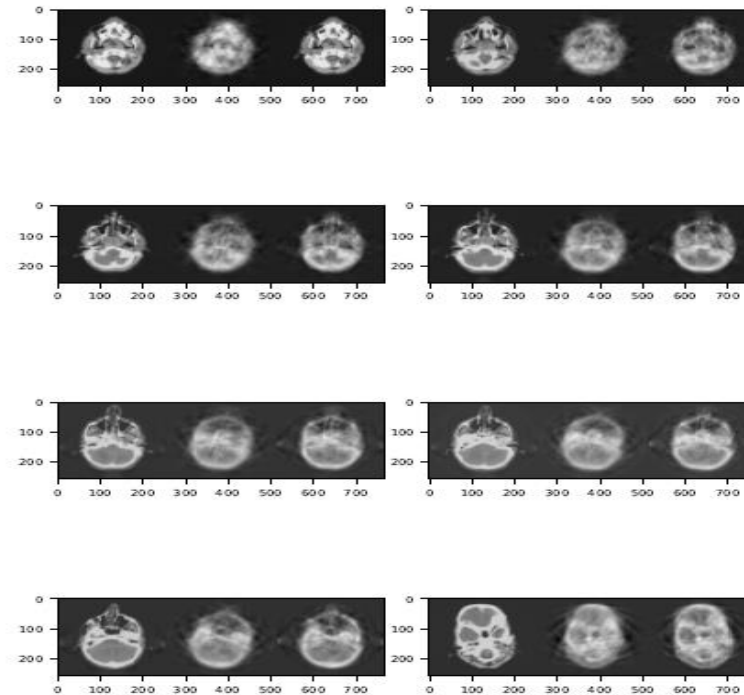


Figure 5.6: Reconstructing from sub-sampled datas. Left original, middle SART method, right kernel extrapolation.

- > 1 Introduction
- > 2 Brief overview of methods of machine learning
- > 3 Kernel and differentiation techniques for machine learning
- > 4 Kernel methods for optimal transportation
- ✓ 5 Supervised Machine Learning applications
  - 5.1 the MNIST problem - supervised learning
  - 5.2 Reconstruction from sub-sampled signals.
- > 6 Unsupervised Machine Learning Applications
- > 7 Optimal transportations applications
- > 8 Partial Differential Equations Applications

# Unsupervised learning methods (clustering / segmentation) a methodology to benchmark them all

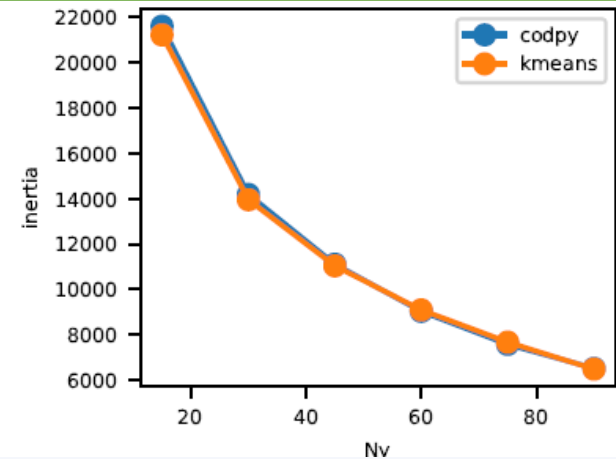
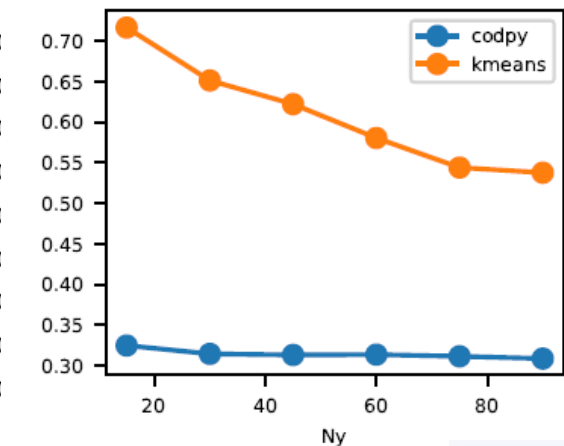
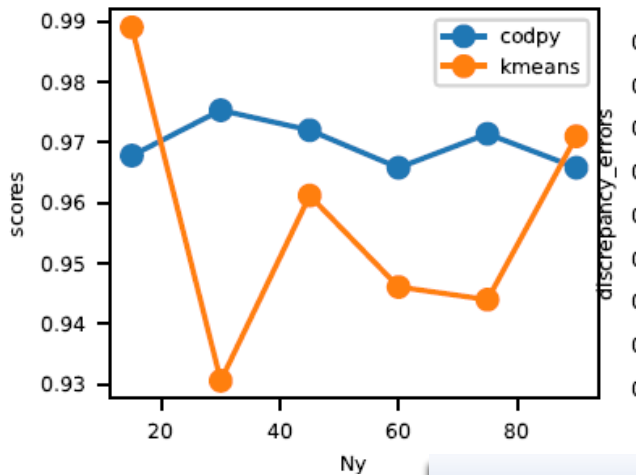
Prediction function  $y = P(x, N_y)$

- $x \in \mathbb{R}^{N_x \times D}$  training set.
- $N_y$  number of clusters.
- $y \in \mathbb{R}^{N_y \times D}$  clusters (segmentation / quantization)

**Benchmark:** k-means algorithm versus sharp discrepancy sequences

**DataBase :** Kaggle credit card fraud detection (284 00 entries / 500 frauds)

**Performance indicators :** scores, discrepancy errors, inertia



**Note:**  
Kernel scores are stables

**Why ? Discrepancy sequences are deterministic**  
K-means clusters are not (initial values).

- > 1 Introduction
- > 2 Brief overview of methods of machine learning
- > 3 Kernel and differentiation techniques for machine learning
- > 4 Kernel methods for optimal transportation
- > 5 Supervised Machine Learning applications
- > 6 Unsupervised Machine Learning Applications
  - 6.1 the MNIST problem - unsupervised learning
  - 6.2 German Credit Risk
  - 6.3 Credit Card Marketing Strategy
  - 6.4 Credit Card Fraud Detection
  - 6.5 Portfolio of Stocks Clustering
- > 7 Optimal transportations applications
- > 8 Partial Differential Equations Applications

# What differentiate kernel methods from other machine learning ?

## Differential operators

RKHS is a theory of functional spaces

Kernel-based learning machines come with  
**Differential / integral operators**

- 1 Introduction
- 2 Brief overview of methods of machine learning
- 3 Kernel and differentiation techniques for machine learning
  - 3.1 Aim of this chapter
  - 3.2 Fundamental notions for supervised learning
  - 3.3 Kernel engineering
  - 3.4 Operations with kernels
  - 3.5 Kernel operators**
  - 3.6 A kernel-based clustering algorithm
- 4 Kernel methods for optimal transportation
- 5 Supervised Machine Learning applications
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

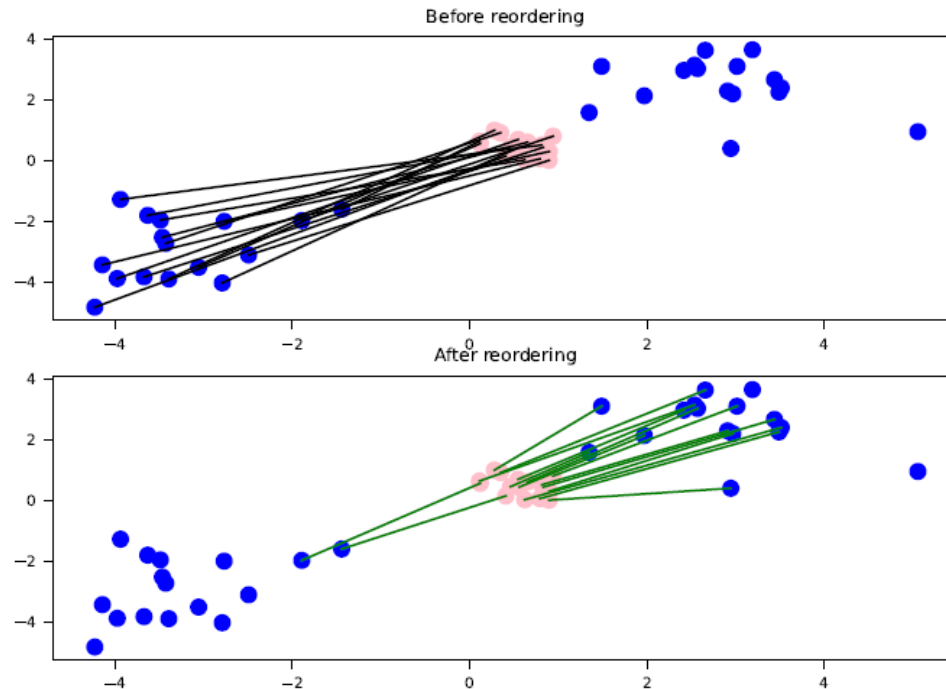
<b>3</b>	<b>Kernels operators</b>	<b>14</b>
3.1	Coefficients operator . . . . .	14
3.2	Partition of unity . . . . .	15
3.3	$\nabla$ operator . . . . .	16
3.4	$\nabla^T$ operator . . . . .	17
3.5	Laplace operator . . . . .	18
3.6	Inverse Laplace operator $\Delta_{k,x,y}^{-1}$ . . . . .	19
3.7	The $\nabla^{-1}$ operator . . . . .	20
3.8	The $(\nabla^T)^{-1}$ operator . . . . .	22
3.9	Leray-orthogonal operator . . . . .	23
3.10	Leray operator and Helmholtz-Hodge decomposition . . . . .	24



Every kernel learning machine has access to  
**OPTIMAL TRANSPORTATION** algorithms

# Optimal Transportation : Monge-Kantorovitch and kernel-based reordering algorithms

$\inf_{\gamma} D_k(x, z) \cdot \gamma, \quad \gamma$  bi-stochastic,  $D_k(x, z)$  discrepancy errors



- 1 Introduction
- 2 Brief overview of methods of machine learning
- 3 Kernel and differentiation techniques for machine learning
- 4 Kernel methods for optimal transportation
  - 4.1 Discrete reordering algorithms
  - 4.2 The conditional expectation algorithm
  - 4.3 The polar factorization algorithm
- 5 Supervised Machine Learning applications
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

# Optimal Transportation : Polar factorization algorithms

**Polar Factorization:** find  $h$  convex s.t.  $z = (\nabla h) \circ T(x)$

**Application :** the sampling function  $z = S_k(x, N_z)$

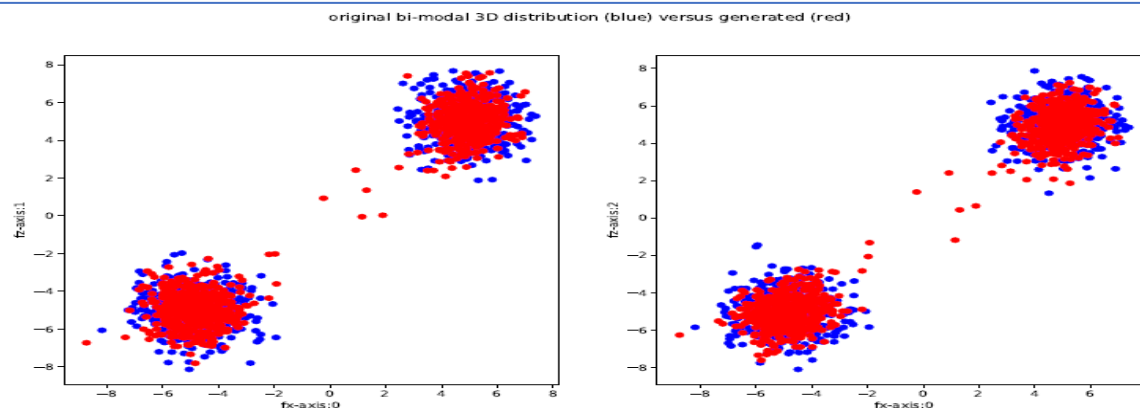
**Inputs:**  $k$  kernel

•  $x \in \mathbb{R}^{N \times D}$  iid of  $X$ .

•  $N_z$  number of samples.

**Output:**  $z \in \mathbb{R}^{N_z \times D}$ ,  
a distribution sharing close statistical properties with  $x$

## Example with a multi-dimensional bi-modal distribution



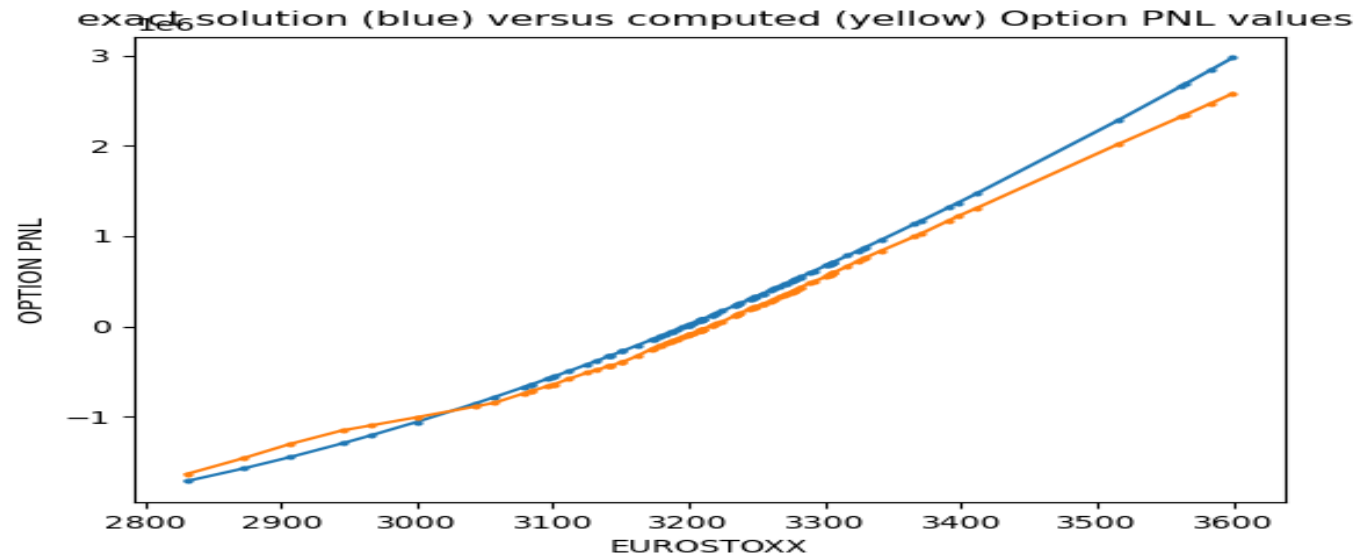
- > 1 Introduction
- > 2 Brief overview of methods of machine learning
- > 3 Kernel and differentiation techniques for machine learning
- ✓ 4 Kernel methods for optimal transportation
  - 4.1 Discrete reordering algorithms
  - 4.2 The conditional expectation algorithm
  - 4.3 The polar factorization algorithm
- > 5 Supervised Machine Learning applications
- > 6 Unsupervised Machine Learning Applications
- > 7 Optimal transportations applications
- 8 Partial Differential Equations Applications



# Polar factorization algorithms applications: reconstruction of time dependent functions from time series observations

## Stress test and reverse stress test

- 1 - Observe a time serie  $X_{t<0}$  (market observations)
- 2 - Observe a time serie  $P\&L(X_{t<0})$  (P&L observations)
- 3 - today =  $t = 0$ . Set  $T > 0$  an Horizon
- 4 - Use  $S_k(X_t, N_z)$  to reproduce the distribution  $P\&L(X_T)$  (stress test)
- 5 - Use  $S_k(P\&L(X_t), N_z)$  to find scenari  $X_T$  (reverse stress test)
- 6 - Use error estimates to produce confidence level on risk measurements



**Note (reverse stress test)**

**discrepancy errors are more meaningful than Mahalanobis distance for kernel methods**

# Polar factorization algorithms applications: Transition probability / conditional expectations

Transition probability operator  $f_{z|x} = \Pi(x, z, f(z), k)$

Consider  $t \mapsto X_t$  any martingale process.

Inputs:  $k$  kernel

- $x \in \mathbb{R}^{N \times D}$  iid of  $X_{t_1}$ .
- $z \in \mathbb{R}^{N \times D}$  iid of  $X_{t_2}$ ,  $t_2 > t_1$ .
- $f(z) \in \mathbb{R}^{N \times D_f}$  optional function values.

Output: transition probabilities operator

- $f_{z|x} \sim \mathbb{E}^{X_{t_2}}(f(\cdot) | X_{t_1} = x) \in \mathbb{R}^{N \times D_f}$ .
- $\Pi(x, z) = p(z^n | x^m)_{n=1..N}^{m=1..N} \in \mathbb{R}^{N \times N}$ .  
stochastic matrix - probabilities of transition.

## Straightforward applications

- Pricing / risk (eg XVA) with internally generated samples
- alternative to Bayesian classifiers

- 1 Introduction
- 2 Brief overview of methods of machine learning
- 3 Kernel and differentiation techniques for machine learning
- 4 Kernel methods for optimal transportation
  - 4.1 Discrete reordering algorithms
  - 4.2 The conditional expectation algorithm
  - 4.3 The polar factorization algorithm
- 5 Supervised Machine Learning applications
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications



# Benchmark of conditional expectations algorithms : the Bachelier problem

Consider  $t \mapsto X_t \in \mathbb{R}^D$  a Brownian process (random parameters).

**Inputs:**  $k$  kernel

- $x \in \mathbb{R}^{N \times D}$  iid of  $X_{t=1}$ .
- $z \in \mathbb{R}^{N \times D}$  iid of  $X_{t=2}$ .
- $f(z) := (z^T a - K)^+$  option payoff,  $a$  random weights

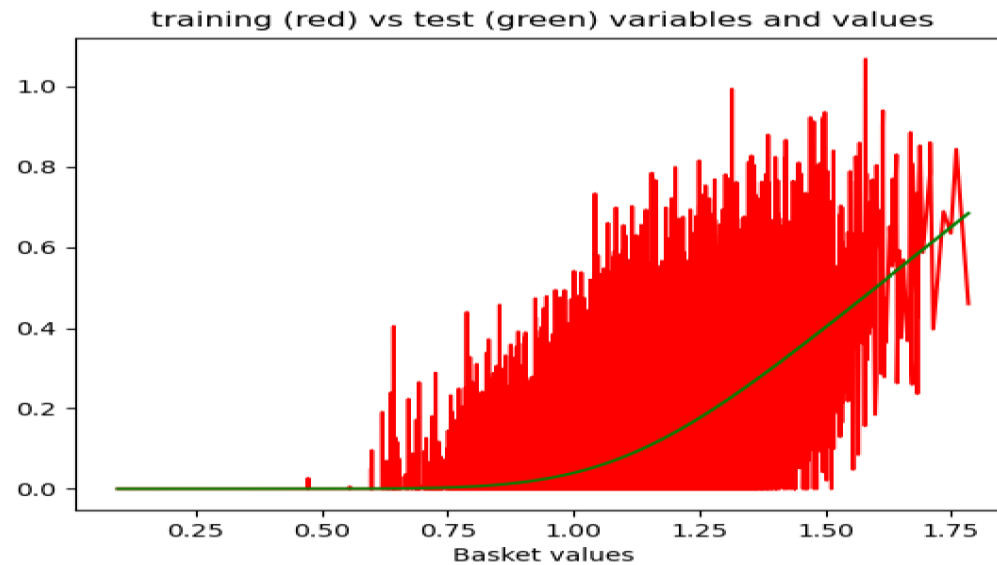
We want to deduce the green curve  $f(z|x)$  from the red data  $f(z)$

**Output:** forward values of options

- $f_{z|x} \sim \mathbb{E}^{X_{t=2}}(f(\cdot)|X_{t=1} = x)$  computed.

benchmarked against

$f(z|x) := \mathbb{E}^{X_{t=2}}(f(\cdot)|X_{t=1} = x)$  - closed formula.



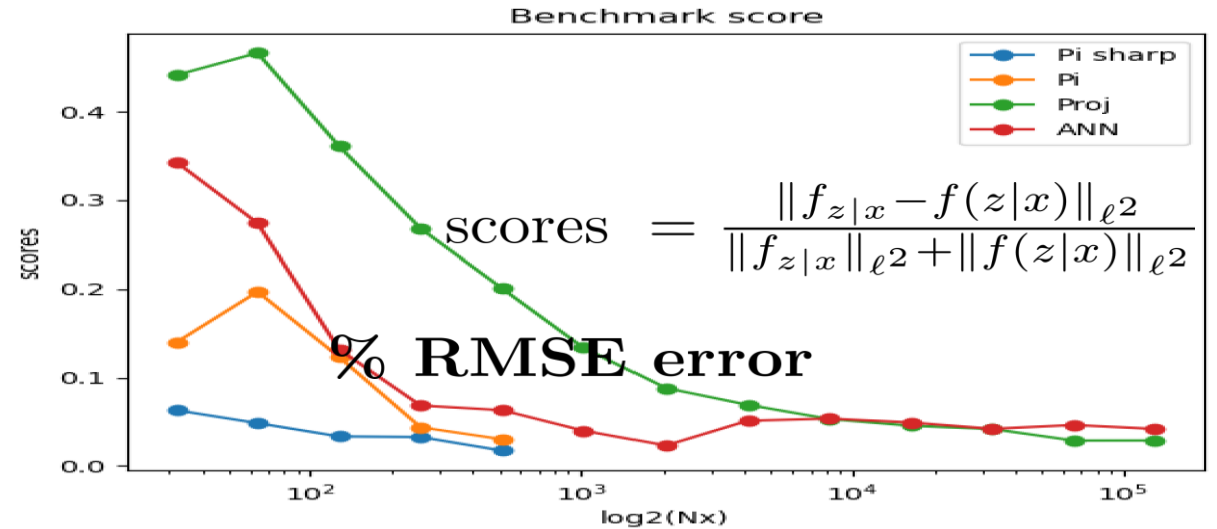
- > 1 Introduction
- > 2 Brief overview of methods of machine learning
- > 3 Kernel and differentiation techniques for machine learning
- > 4 Kernel methods for optimal transportation
- > 5 Supervised Machine Learning applications
- > 6 Unsupervised Machine Learning Applications
- ✓ 7 Optimal transportations applications
  - 7.1 the Bachelier problem
- > 8 Partial Differential Equations Applications

# Kernel methods - Tools for statistical Learning

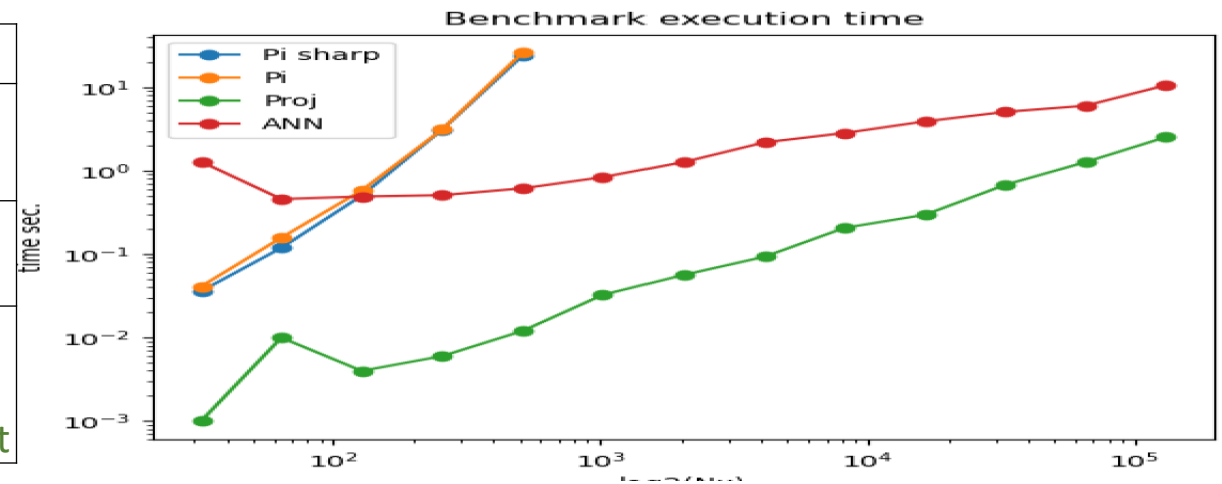
## the Bachelier problem : a toy example of audit

### Score and time Benchmark :

- **ANN = Tensorflow**
- **Proj = CodPy P(x,y,z,fx,k)**
- **Pi = CodPy Pi(x,z,fx,k)**
- **Pi sharp = CodPy Pi(x,z,fx,k)**  
(x,z sharp discrepancy sequences)



method	Sharp	Pi	ANN	Proj
size of training set	64	512	100 000	100 000
error % RMSE	3.40%	3.00%	4.10%	2.80%
diagnostic	convergent	convergent	NOT convergent	NOT convergent



# Optimal Transportation :

## Signed Polar factorization algorithms and the convex-hull algorithms (CHA) explicit solutions to Hamilton-Jacobi equations

### CHA algorithm:

- Decompose  $z = (\nabla h) \circ T(x)$ ,  $T$  "positive".
- Consider  $z^+ = (\nabla h^+) \circ T(x)$ ,  $h^+$  convex hull of  $h$ .

- > 1 Introduction
- > 2 Brief overview of methods of machine learning
- > 3 Kernel and differentiation techniques for machine learning
- ✓ 4 Kernel methods for optimal transportation
  - 4.1 Discrete reordering algorithms
  - 4.2 The conditional expectation algorithm
  - 4.3 The polar factorization algorithm
- > 5 Supervised Machine Learning applications
- > 6 Unsupervised Machine Learning Applications
- > 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

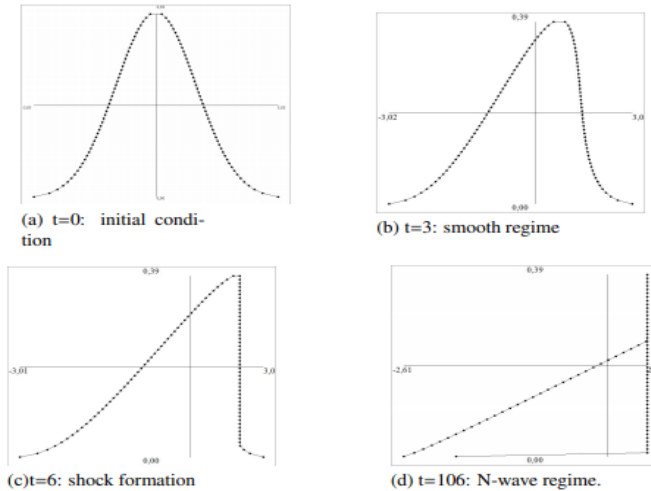
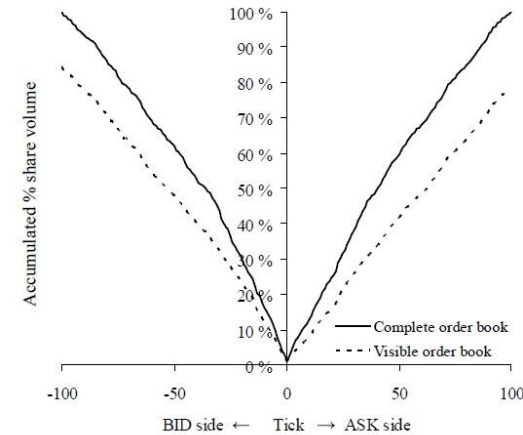
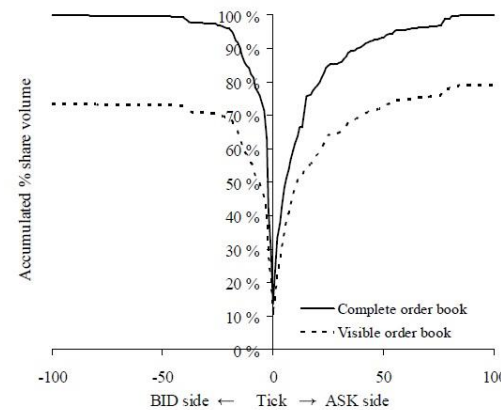


Figure 4.1: Entropy dissipative solutions to Burgers equation



# What differentiate kernel methods from other machine learning ?

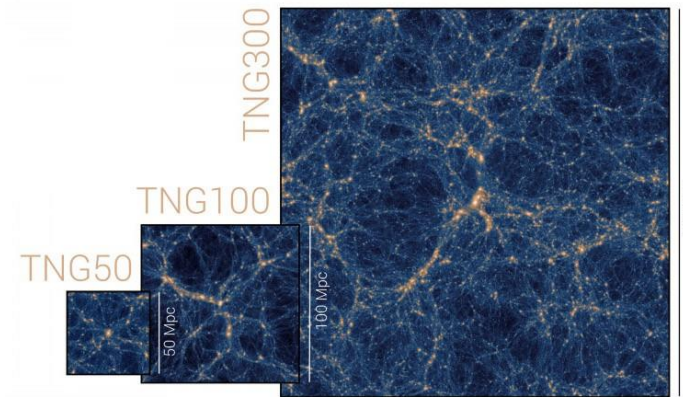
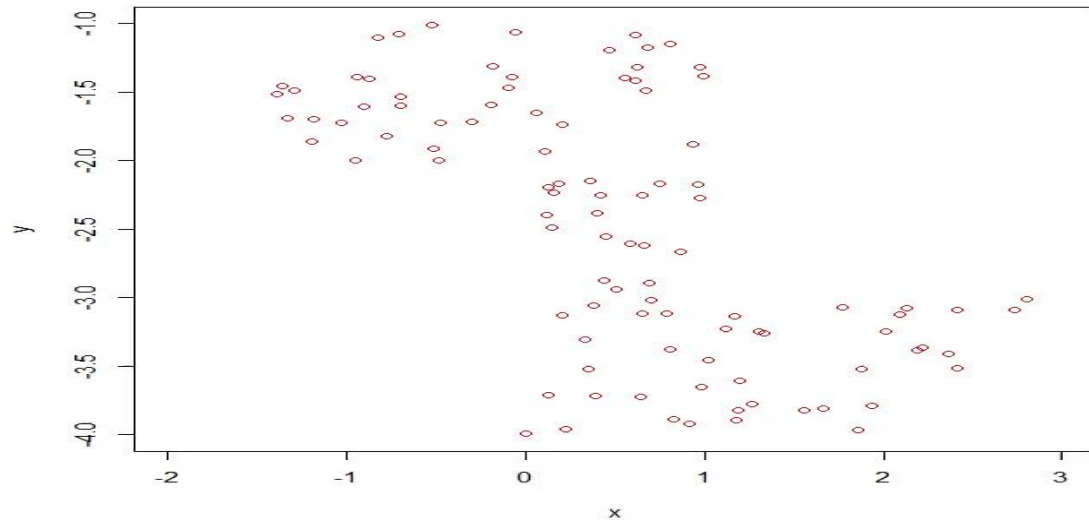
## Differential operators / sharp discrepancy sequences

### Transported meshfree methods

Share similarities with  
**smooth particle hydrodynamic**

Fichier Historique Redimensionnement

time= 0.14 max = 6.731



**Apps:**  
**Video games**  
**Astronomical simulations**

...

- 1 Introduction
- 2 Brief overview of methods of machine learning
- 3 Kernel and differentiation techniques for machine learning
- 4 Kernel methods for optimal transportation
- 5 Supervised Machine Learning applications
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

# What are kernel methods for numerical simulations in Finance ?

High-dimensional **Hamilton-Jacobi-Bellman** PDE solvers

Front Office / Risk management engines **since 2012**

$$dX_t = r(t, X_t)dt + \sigma(t, X_t)dW_t$$
$$P(t, X_t, \nabla g(X_t), \dots)$$

Any kind of stochastic process, any dimensions

Any kind of function (payoff / strategy  
optimal control)

1. Any kind of forward / backward modeling (**BSDE/FSDE**).
2. **PDE** (Partial Differential Equation) solver engine  
**Fokker-Plank / Kolmogorov (Forward / Backward)**
3. Huge portfolios / risks sources capabilities.
4. Bullet proof **auditable** due to **worst-error bounds**.
5. Any kind of risk measurements (from instant prices to **forward greeks**).
6. **Real time** versus **accuracy** easy tuning capabilities.

**Apps:**  
**Pricing / Risk / XVA**  
**/ investment**  
**strategies /**  
**arbitrage / etc etc**  
...

- 1 Introduction
- 2 Brief overview of methods of machine learning
- 3 Kernel and differentiation techniques for machine learning
- 4 Kernel methods for optimal transportation
- 5 Supervised Machine Learning applications
- 6 Unsupervised Machine Learning Applications
- 7 Optimal transportations applications
- 8 Partial Differential Equations Applications

## Want to dig in ? Our bibliography

### theoretical

We propose novel methods for machine learning and numerical simulations, using a partial differential equations approach.

- 1) [A class of mesh-free algorithms for mathematical finance, machine learning and fluid dynamics](#). This paper is the backbone of our approach.
- 2) "[The Transport-based Mesh-free Method \(TMM\) and its applications in finance: a review](#)", in [Wilmott magazine](#), This paper is a general, high-level description of our approach.
- 3) "[Mesh-free error integration in arbitrary dimensions: A numerical study of discrepancy functions](#)", [Computer Methods in Applied Mechanics and Engineering](#). This paper focuses on error analysis for Reproducing Kernel Hilbert Space methods.
- 4) « [A new method for solving Kolmogorov equations in mathematical finance](#) », *Comptes Rendus Mathematique*, Volume 355, Issue 6, June 2017, Pages 680-686. This paper explains the Partial Differential Equation strategy for mathematical finance and provides numerical examples.
- 5) "[Revisiting the method of characteristics via a convex hull algorithm](#)" *Journal of Computational Physics*, October 2015 <https://doi.org/10.1016/j.jcp.2015.05.043> applies this method for conservation laws.
- 6) [A high dimensional framework for financial instruments valuation](#), 2013 an early attempt to describe multidimensional PDE for mathematical finance.
- 7) [Optimally transported schemes](#) 2008. treat the one dimensional case.

### Algorithmic / numerical

We provide a library, named CodPy, which stands for "Curse of dimensionality in Python". It provides tools for machine learning, statistical learning, numerical simulations, and is based on our understanding of RKHS methods.

- 1) [Codpy Tutorial](#) is a gentle introduction to this library, focusing on machine learning.
- 2) [CodPy - Advanced Tutorial](#), is a technical description of this library.
- 3) [A kernel based reordering algorithm](#) describes a central reordering algorithm for our application.
- 4) *A kernel based polar factorization and the sampling algorithm with CodPy. In preparation.* describes an algorithm to compute polar factorization. This algorithm is illustrated with a very handy tool, allowing to produce iid samples from any input distribution.
- 5) *A kernel based algorithm to compute conditional expectations. In preparation.* is a quite important algorithm for finance applications. We benchmark an implementation of this algorithm, using our framework CodPy, against a classical neural network one.
- 6) [Hedging Strategies for Net Interest Income and Economic Values of Equity](#) describes a prototype using CodPy, aiming to build sophisticated strategies for ALM purposes.
- 7) [Kernel methods for stress test and reverse stress test](#) is a support vector machine approach to these class problems, using CodPy.
- 8) [Numerical results using codofi](#) collects a number of academical tests for pricing purposes with our approach