Topics in Financial Machine Learning

8th Annual Bloomberg Columbia Machine Learning in Finance Workshop

Dr Paul A. Bilokon

Imperial College London / Thalesians Ltd

2022.09.23

Iterated and exponentially weighted moving principal component analysis

David Finkelstein
DQR Ltd

The principal component analysis (PCA) is a staple statistical and unsupervised machine learning technique in finance. The application of PCA in a financial setting is associated with several technical difficulties, such as numerical instability and nonstationarity. We attempt to resolve them by proposing two new variants of PCA: an iterated principal component analysis (IPCA) and an exponentially weighted moving principal component analysis (EWMPCA). Both variants rely on the Ogita–Aishima iteration as a crucial step.

Background

- The principal component analysis (PCA) [Jol02, JC16] invented by Pearson [Pea01] and improved by Hotelling [Hot33, Hot36] is a staple statistical and unsupervised machine learning technique in finance [AA21].
- Its central idea is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set.
- This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables.

Technical difficulties

- The entire data set may not be immediately available (it may be arriving piecewise in real time), so one is forced to work with its subsets pertaining to different time intervals.
- First, when the PCs are computed separately on each subset, the geometry of the resulting PCs may suffer from numerical artifacts. In particular, the sign of a given PC may “flip” from one subset to the next.
- Second, financial data are rarely stationary, and the assumption of a constant covariance matrix is rarely justified.

Proposed solutions

- To remedy the first problem, we propose an iterated principal component analysis (IPCA): instead of computing the principal components on each arriving subset independently, we iteratively refine them from one subset to the next.
- To remedy the second problem, we combine the aforementioned iterative refinement with an exponentially weighted moving computation of the covariance matrix, to obtain an exponentially weighted moving principal component analysis (EWMPCA).
- We are heavily indebted to Ogita and Aishima, who proposed an iterative refinement method for symmetric eigenvalue decomposition [OA18], on whose work we build.

Data sets

- We have used two data sets in this study. Both are derived from data supplied by FirstRate Data and both consist of hourly returns on futures.
- The first data set (Data Set 1) covers the period 20th August, 2007 to 4th June, 2021, both inclusive, and consists of hourly returns on equity futures: DAX (DY), E-Mini S&P 500 (ES), E-Mini S&P 500 Midcap (EW), Euro Stoxx 50 (FX), CAC40 (MX), E-Mini Nasdaq-100 (NQ), E-Mini Russell 2000 (RTY), FTSE 100 (X), Dow Mini (YM).
- The second data set (Data Set 2) covers the period 10th September, 2012 to 4th June, 2021, both inclusive, and consists of hourly returns on fuel futures: Brent Last Day Financial (BZ), Crude Oil WTI (CL), Natural Gas (Henry Hub) Last-day Financial (HH), NY Harbor ULSD (Heating Oil) (HO), Henry Hub Natural Gas (NG), RBOB Gasoline (RB).

Classical PCA: the formalism

- A data set of n observations of p features can be represented by an $n \times p$ data matrix X, whose jth column, $x_{:,j}$, is the vector of n observations of the jth feature.
- We seek a linear combination of the columns of matrix X with maximum variance. Such linear combinations are given by $Xw^{(1)}$, where $w^{(1)}$ is a vector of constants $w_1^{(1)}, \ldots, w_p^{(1)}$.
- It can be shown that $w^{(1)}$ must be a (unit-norm) eigenvector of the sample covariance matrix Q associated with the data set; more precisely, the eigenvector corresponding to the largest eigenvalue $\lambda^{(1)}$ of Q.
- The full set of eigenvectors $w^{(1)}, \ldots, w^{(p)}$ of Q, corresponding to the eigenvalues sorted in decreasing order, $\lambda^{(1)}, \ldots, \lambda^{(p)}$, are the solutions to the problem of obtaining up to p linear combinations $Xw^{(k)}$, $1 \leq k \leq p$, which successively maximize variance, subject to uncorrelatedness with previous linear combinations.
- We call these linear combinations the principal components (PCs) of the data set.

Classical PCA: the implementation details

- It is standard to define PCs as the linear combinations of the centred variables $x^*_{:,j}$, with generic element $x^*_{ij} = x_{ij} - \bar{x}_{:,j}$, where $\bar{x}_{:,j}$ denotes the mean value of the observations on variable j.
- At the core of PCA is the eigendecomposition of the sample covariance matrix Q or, equivalently, the singular value decomposition (SVD) of the data matrix X.
- An industry-standard implementation of PCA is `sklearn.decomposition.PCA` in the software library scikit-learn [PVG+11]. It uses the LAPACK [ABB+99] implementation of the full SVD or a randomized truncated SVD by the method of Halko *et al.* [HMT11].
- When comparing our results to the classical PCA it is this implementation that we use as a benchmark.

The Ogita–Aishima algorithm

- Ogita and Aishima have proposed and analyzed an iterative refinement algorithm [OA18, Algorithm 1] for approximate eigenvectors $\hat{X}$ of A.
- The authors demonstrate the monotone and quadratic convergence of the algorithm under some reasonable technical conditions.

1 **function** OGITA_AISHIMA_STEP(A, $\hat{X}$)
2 $\quad\tilde{\lambda}, R, S \leftarrow$ ESTIMATE_EIGENVALUES(A, $\hat{X}$, **true**) ▷ Compute approximate eigenvalues.
3 $\quad\tilde{D} \leftarrow ({}^{(\mathrm{diag})}\tilde{\lambda}_i)$
4 $\quad\delta \leftarrow 2(||S - \tilde{D}||_2 + ||A||_2||R||_2)$
5 $\quad$**for** $i \leftarrow 1$ to n **do**
6 $\quad\quad$**for** $j \leftarrow 1$ to n **do**
7 $$\tilde{e}_{ij} \leftarrow \begin{cases} \frac{s_{ij}+\tilde{\lambda}_j r_{ij}}{\tilde{\lambda}_j - \tilde{\lambda}_i}, & \text{if } |\tilde{\lambda}_i - \tilde{\lambda}_j| > \delta; \\ r_{ij}/2, & \text{otherwise.} \end{cases}$$ ▷ Compute $\tilde{E}$.
8 $\quad X' \leftarrow \hat{X} + \hat{X}\tilde{E}$
9 $\quad$**return** X'

Estimating the eigenvalues

- We need to be able to estimate the eigenvalues of a given real symmetric matrix $A \in \mathbb{R}^{n \times n}$ for a precomputed set of eigenvectors $\hat{X} \in \mathbb{R}^{n \times n}$ (the eigenvectors are in the columns of $\hat{X}$):

```
1 function ESTIMATE_EIGENVALUES(A, X̂, return_extra)
2     R ← I − X̂ᵀ X̂
3     S ← X̂ᵀ A X̂
4     for i ← 1 to n do
5         λ̃_i ← s_ii / (1 − r_ii)
6     if return_extra then
7         return λ̃, R, S
8     else
9         return λ̃
```

Achieving the required level of convergence

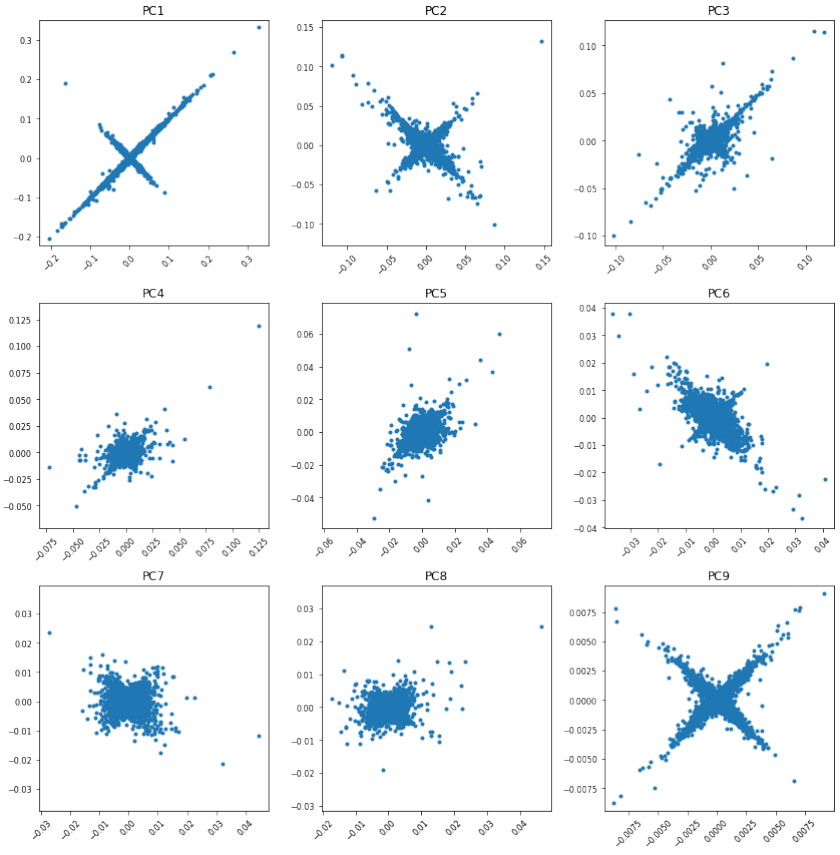
- We wrap the function `ogita_aishima_step` in a higher-level function that performs the number of iterations required for satisfying a sensible convergence criterion.

```
1  function OGITA_AISHIMA(A, X̂, tol=1e-6, max_iter_count=none, sort_by_eigenvalues=false)
2      iter_count ← 0
3      while true do
4          iter_count ← iter_count + 1
5          X̂′ ← OGITA_AISHIMA_STEP(A, X̂)
6          if max_iter_count is not none and iter_count == max_iter_count then
7              break
8          ϵ ← ||X̂′ − X̂||_2
9          if ϵ ¡ tol then
10             break
11         X̂ ← X̂′
12     if sort_by_eigenvalues then
13         λ̃ ← ESTIMATE_EIGENVALUES(A, X̂′, false)
14         Sort λ̃ in descending order and reorder the corresponding columns of X̂′ to
   match that order
       return X̂′
```

Iterated PCA

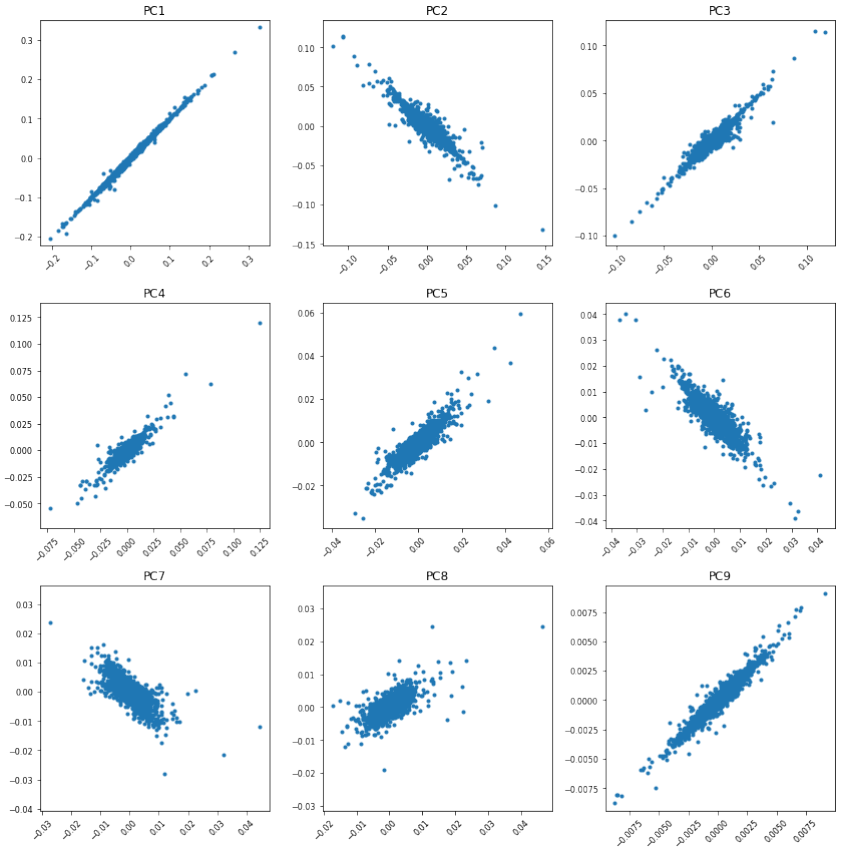
- Iterated PCA (IPCA) is a straightforward extension of PCA wherein the algorithm can be fitted multiple times.
- Every time fit is invoked on a new data subset, that subset's sample covariance matrix Q is calculated.
- The eigenvectors $\hat{W}$ of Q are stored between the fits; for each new Q the previous eigenvectors are used as an initial guess in `ogita_aishima`(Q, $\hat{W}$, `sort_by_eigenvalues=`**true**).
- At the beginning, when no initial guess is available, the eigenvectors are obtained using standard methods.

Classical PCA



We perform classical PCA on Data Set 1 twice. In the first instance, we perform PCA on the entire dataset. In the second, we perform PCA on each year individually and stack the results together. We then produce scatter plots, where the second result (y-axis) is plotted against the first result (x-axis).

IPCA



We perform classical PCA on Data Set 1. Then we perform IPCA on each year individually and stack the results together. We then produce scatter plots, where the second result (y-axis) is plotted against the first result (x-axis). IPCA resolves the numerical instability problem witnessed on the previous slide.

Moving statistics

- Let $x_1, \ldots, x_t$, $t \in \mathbb{N}$, be a sequence of p-dimensional observations.
- The exponentially weighted moving average for this sequence can be calculated recursively as

$$m_t = \begin{cases} x_1, & t = 1, \\ (1-\alpha)x_t + \alpha m_{t-1}, & t > 1, \end{cases}$$

  where $0 < \alpha < 1$ is a constant parameter.
- Tsay [Tsa10] proposes a similar moving statistic for the sample covariance:

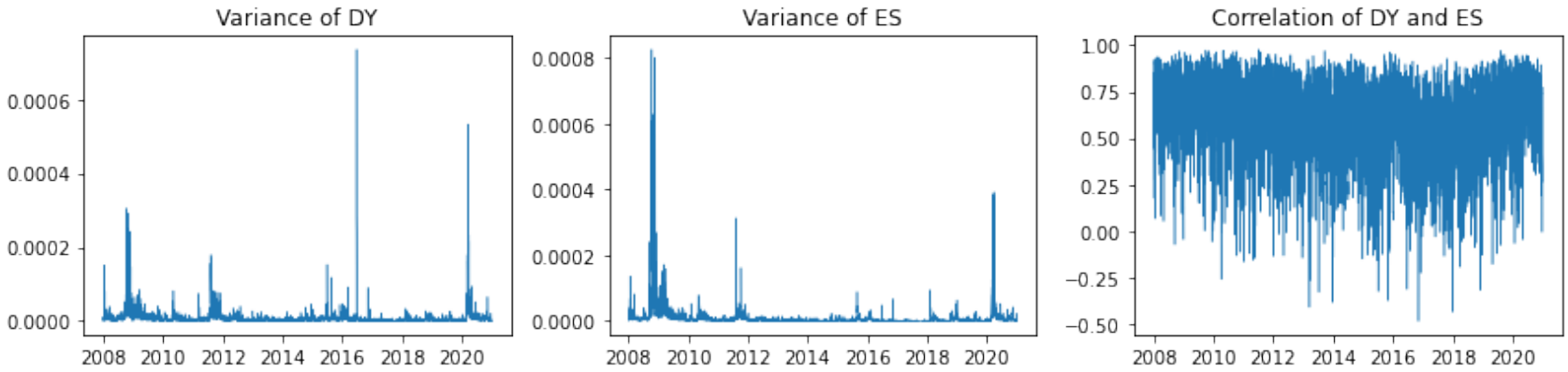
$$S_t = \begin{cases} 0_{p\times p}, & t = 1, \\ (1-\alpha)(x_t - m_t)(x_t - m_t)^\intercal + \alpha S_{t-1}, & t > 1, \end{cases}$$

- One way to estimate the parameter α is by using maximum likelihood (ML). For example, if $x_1, \ldots, x_T$, $t \in \mathbb{N}$, are normally distributed, then α_{ML} is the value of α that maximizes

$$\ln \mathcal{L}(\alpha) \propto -\frac{1}{2}\sum_{t=1}^{T} |S_t| - \frac{1}{2}\sum_{t=1}^{T} (x_t - m_t)^\intercal S_t^{-1} (x_t - m_t).$$

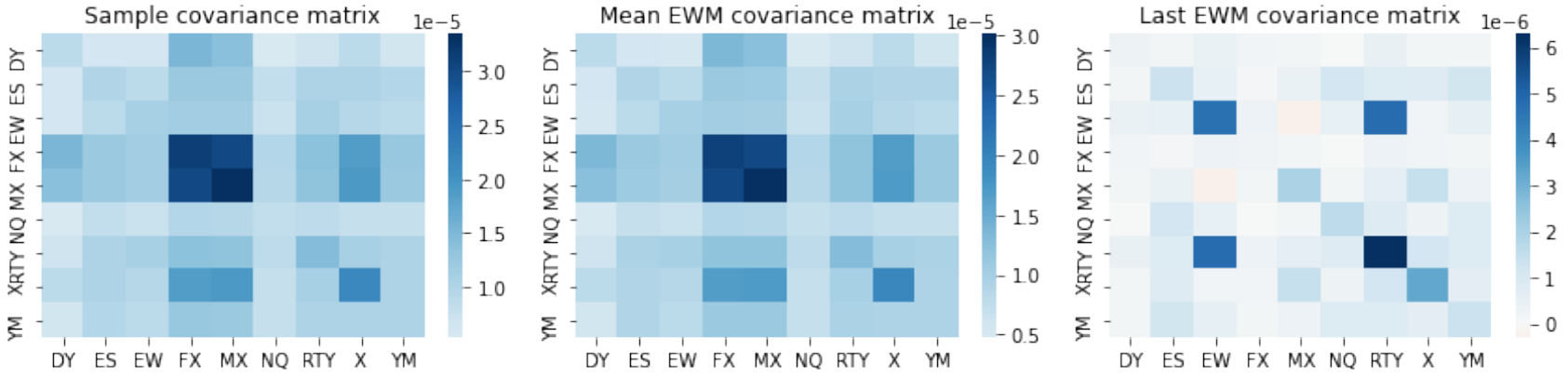
- In an example in Section 10.1 of [Tsa10], Tsai describes the value $\alpha \approx 0.9305$ as being in the typical range commonly seen in practice.

Nonstationarity



Moving statistics reveal the nonstationary nature of financial data. Consider Data Set 1 as an example. The variances of the returns on individual futures change over time and exhibit the so-called volatility clustering [Con07]; the correlations between pairs of futures are also time-varying.

The covariance matrix is time-varying



- Heatmaps comparing the sample covariance matrix with a time series of exponentially weighted covariance matrices.
- Whereas the mean exponentially weighted moving covariance matrix resembles the sample covariance matrix, individual exponentially weighted moving covariance matrices (such as the last one in our time series shown above) may differ from it significantly.
- The principal components obtained using the sample covariance matrix present an averaged picture; we need a more precise tool to work out what's going on at each time step.

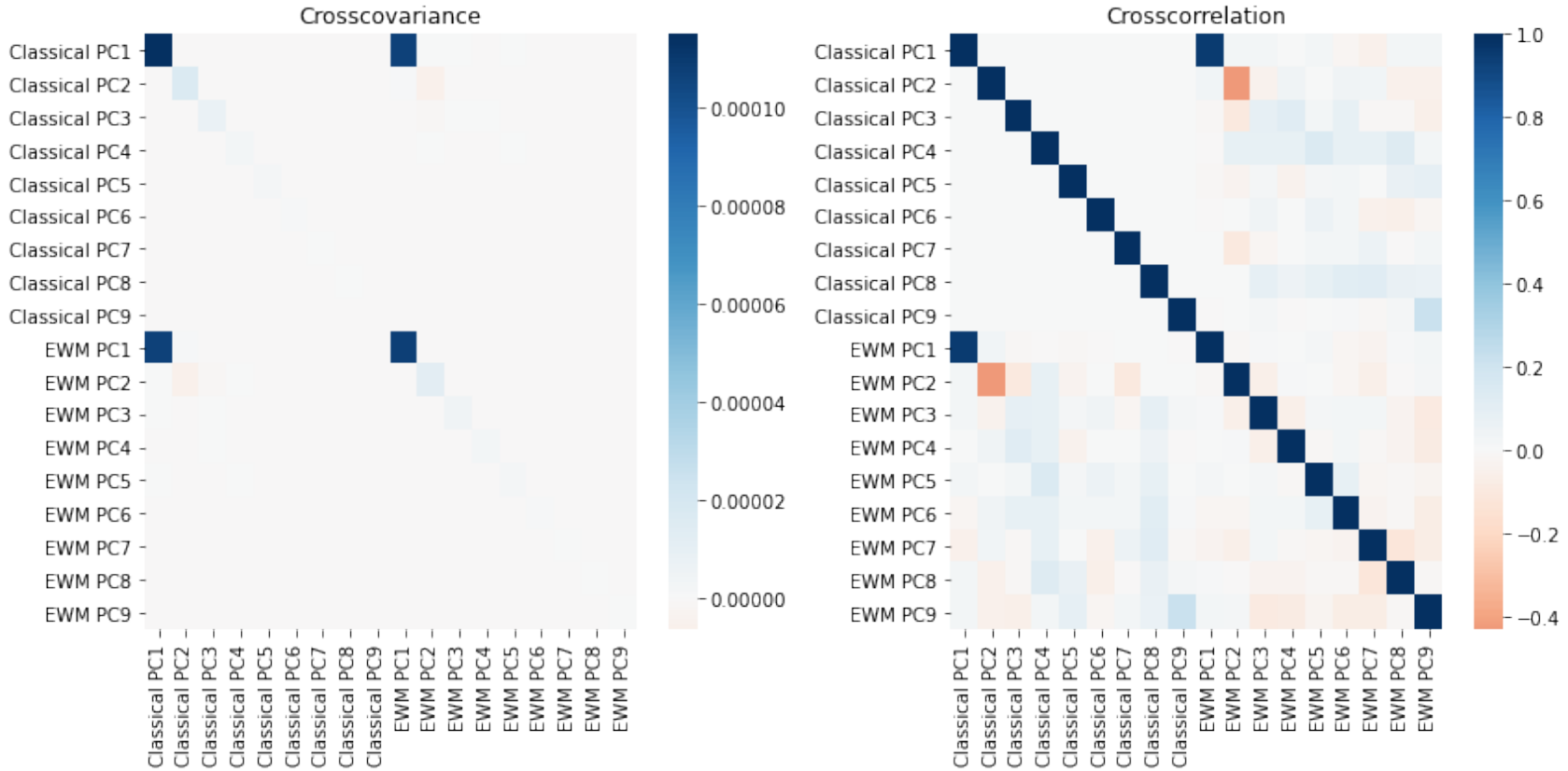
Exponentially weighted moving PCA

- Combining ideas from the Ogita–Aishima iteration and moving statistics it is straightforward to formulate an exponentially weighted moving PCA (EWMPCA).

function EWMPCA(X, α, $\hat{W}^{\text{initial}}$, tol=1e-6, max_iter_count=**none**)
  for $i = 1, \ldots, n$ **do**
    if $i == 1$ **then**
      $m \leftarrow x_{i,:}^{\mathsf{T}}$ ▷ Exponentially weighted moving average.
      $S \leftarrow 0_{p \times p}$ ▷ Exponentially weighted moving covariance.
      $\hat{W} \leftarrow \hat{W}^{\text{initial}}$ ▷ Eigenvectors of S.
      $z_{i,:} \leftarrow 0_{1 \times p}$ ▷ Principal components.
    else
      $m \leftarrow (1 - \alpha) x_{i,:}^{\mathsf{T}} + \alpha m$
      $x^* \leftarrow x_{i,:}^{\mathsf{T}} - m$
      $S \leftarrow (1 - \alpha) x^* (x^*)^{\mathsf{T}} + \alpha S$
      $\hat{W} \leftarrow$ OGITA_AISHIMA(S, $\hat{W}$, tol, max_iter_count, sort_by_eigenvalues=**true**)
      $z_{i,:} \leftarrow (x^*)^{\mathsf{T}} \hat{W}$
  return z

- $\hat{W}^{\text{initial}}$ must be such as to facilitate convergence. One option is to use the sample covariance matrix for the first few (say 100) observations.

PCA vs EWMPCA



Heatmaps of the crosscovariance and crosscorrelation between the classical PCA and EWMPCA principal components computed on Data Set 1. As we can see, the EWMPCA principal components are not pairwise uncorrelated; by construction, they are uncorrelated locally, not on average. However, the pairwise correlations are low. For the most part, the EWMPCA principal components are distinct from the corresponding classical PCA principal components.

Economic validation

- Has EWMPCA economic significance over and above that of the classical PCA?
- While there are many ways to explore this question, we focus on a particular approach. Avellaneda and Lee have demonstrated in [AL10] that PCA can be used to generate profitable trading strategies. Can EWMPCA better them?
- For each component, we obtain a trading strategy, and a backtest gives us its Sharpe ratio [Sha94].
- We compute the Sharpe ratios for the strategies based on the classical PCA as well as for the strategies based on EWMPCA, while keeping all parameters equal between the two methods.

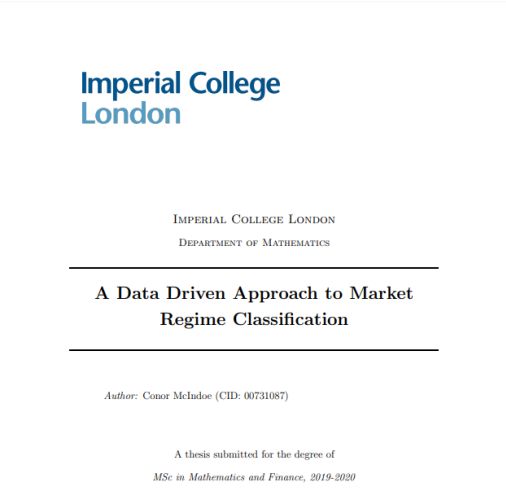
Results

	Data Set 1		Data Set 2	
Principal component	Classical PCA	EWMPCA	Classical PCA	EWMPCA
PC1	**0.65**	0.73	-0.32	0.02
PC2	0.43	**1.02**	-0.39	-0.34
PC3	-0.2	0.89	-0.13	**0.48**
PC4	-0.11	-0.11	-0.47	0.26
PC5	0.5	-0.33	0.3	-0.39
PC6	-0.01	-0.13	**0.37**	-0.16
PC7	-0.5	0.04		
PC8	-0.03	0.06		
PC9	-0.08	0.23		

Implementation

- The code behind this paper is publicly available on GitHub: `https://github.com/sydx/xpca`
- The repository contains a general-purpose Python library, `xpca.py`, and the notebooks that were used to produce the figures in this paper.
- The class `IPCA` implements the iterated PCA algorithm. It has been modelled on `sklearn.decomposition.PCA`, so that `IPCA` can be a drop-in replacement for the former.
- No attempt to achieve industrial-grade performance has been made; in particular, the functions `estimate_eigenvalues`, `ogita_aishima_step`, and `ogita_aishima` could benefit from further optimization.
- The class `EWMPCA`, as the name suggests, implements the EWMPCA algorithm. It can be used in two modes (and the modes can be interleaved):
  - the online mode, where the method `add` is applied to a single observation and returns the corresponding observation transformed to the principal component space;
  - the batch mode, where the method `add_all` is applied to a matrix whose rows are p-dimensional observations; the result is, then, a matrix of principal components.

A Data Driven Approach to Market Regime Classification


Imperial College London

Imperial College London
Department of Mathematics

A Data Driven Approach to Market Regime Classification

Author: Conor McIndoe (CID: 00731087)

A thesis submitted for the degree of
MSc in Mathematics and Finance, 2019-2020

Conor McIndoe. *A Data Driven Approach to Market Regime Classification*. A thesis submitted for the degree of MSc in Mathematics and Finance, 2019-2020.

> *We provide a novel algorithm which attempts to classify market regimes in U.S. equities time series. As far as possible, manual intervention is avoided, preferring a data-driven approach. The path signature is utilized as a central tool; the application of which is justified. We discuss the connection between market regimes and distributions of path signatures, and provide a metric space structure on the latter which allows for a clustering to be formulated. The code both to reproduce and to develop further the clustering algorithms presented is provided on GitHub.*

Paper in preparation; joint work with **Antoine (Jack) Jacquier**.

Distributional Reinforcement Learning for Optimal Execution

Toby Weston. *Distributional Reinforcement Learning for Optimal Execution*. A thesis submitted for the degree of MSc in Mathematics and Finance, 2019-2020.

When trading a financial asset, large orders will often incur higher execution costs as the trader uses up the available liquidity. To reduce this effect, orders are split and executed over a short period of time. Theoretical solutions for how to optimally split orders rely on models of market environments. These fail to take into account market idiosyncrasies and tend to oversimplify a complex optimisation problem.

Deep Q learning provides a set of methodologies for learning an optimal solution from real experience. Successful application would allow models of the trading environment to be sidestepped in favour of direct interaction with the financial markets. Deep Q learning has previously been applied to the problem of optimal execution and has shown promise, both in simulated environments and on historical data.

In the last few years many improvements have been suggested for the vanilla deep Q learning algorithm. Distributional reinforcement learning in particular has shown to outperform value based deep Q learning on a selection of Atari games. Given the highly stochastic nature of the trading environment it is reasonable to assume that it would perform well for the problem of optimal execution.

In the following work we will outline the principles behind distributional reinforcement learning and show that it can outperform value based deep Q learning for optimal execution. To the best of our knowledge this is the first time distributional reinforcement learning has been used for optimal execution.

Toby Weston
Imperial College London

Paper in preparation; joint work with **Antoine (Jack) Jacquier**.


Irene Aldridge and Marco Avellaneda.
Big Data Science in Finance.
Wiley, 2021.


E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen.
LAPACK Users' Guide.
Society for Industrial and Applied Mathematics, Philadelphia, PA, 3rd edition, 1999.


Marco Avellaneda and Jeong-Hyun Lee.
Statistical arbitrage in the US equities market.
Quantitative Finance, 10(7):761–782, August 2010.


Rama Cont.
Volatility clustering in financial markets: Empirical facts and agent-based models.
In *Long Memory in Economics*, pages 289–309. Springer Berlin Heidelberg, 2007.


N. Halko, P. G. Martinsson, and J. A. Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Review, 53(2):217–288, January 2011.


Harold Hotelling.
Analysis of a complex of statistical variables into principal components.
Journal of Educational Psychology, 24(6 and 7):417–441 and 498–520, 1933.


Harold Hotelling.

Relations between two sets of variates.
Biometrika, 28(3/4):321, December 1936.


Ian T. Jolliffe and Jorge Cadima.
Principal component analysis: a review and recent developments.
Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065):20150202, April 2016.


I.T. Jolliffe.
Principal Component Analysis.
Springer, 2nd edition, 2002.


Takeshi Ogita and Kensuke Aishima.
Iterative refinement for symmetric eigenvalue decomposition.
Japan Journal of Industrial and Applied Mathematics, 35:1007–1035, 2018.


Karl Pearson.
On lines and planes of closest fit to systems of points in space.
Philosophical Magazine, 2(11):559–572, 1901.


F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12:2825–2830, 2011.


William F. Sharpe.
The Sharpe ratio.

The Journal of Portfolio Management, 21(1):49–58, October 1994.

Ruey S. Tsay.
Analysis of Financial Time Series.
Wiley Series in Probability and Statistics. Wiley, 2010.