Unsupervised Learning

Silèye Ba

Adjunct Lecturer, Neoma Business School, Paris
Machine Learning Scientist, L'Oreal, Paris

22.02.2023

Overview

1. Reminders about previous sessions

2. Unsupervised learning
   - Clustering
     - Clustering with K-means
     - Kernel K-means
   - Density estimation
     - Non parametric kernel densities
     - Mixture model densities
   - Dimensionality reduction
     - Principal component analysis (PCA)
     - Kernel Principal component analysis (PCA)
     - Neural embeddings

Overview

Questions

About previous sessions

Do you have questions or comments about previous sessions

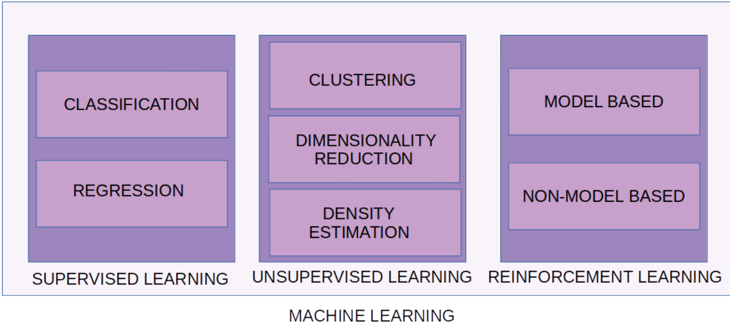
About your projects

- Group project deadline is the **31 of March 2021**
- Do you have questions about the projects' topics
- Personal project will be shared next week in Neoma/Courses
  - You have will two weeks to complete and deliver your work

Projects expectations

- Use what you learned during this class : test many models
- Evaluate your proposals properly : multi-fold cross-validation
- Be creative : it can give extra points

Machine learning approaches overview



Supervised learning methods (last session)

- K-nearest-neighbours : simplicity
- Decision trees and random forests : interpretability
- Kernel methods and support vector machines : solution uniqueness
- Neural networks : efficiency on very large datasets

No single method systematically outperforms : test to find out the best for your problem

Overview

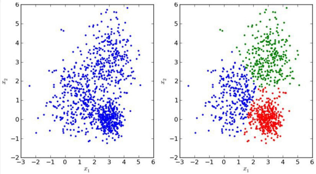
Unsupervised learning

We are given sample data $(\mathbf{x}_n)$, without any labels. The goal is to discover hidden data structure. Unsupervised learning is also called knowledge discovery

Canonical examples

Given data samples $\mathbf{x}_1, ...\mathbf{x}_n, ..., \mathbf{x}_N$

- **Clustering** : cluster samples into K groups
- **Density estimation** : estimate data density $f_{\mathbf{w}}(\mathbf{x}_n) = p(\mathbf{x}_n|\mathbf{w})$
- **Dimensionality reduction** : find lower dimensional embedding space



Unsupervised learning evaluation

- Contrarily to supervised learning, unsupervised learning evaluation is not straightforward
  - There are no associated labels
- Evaluation possible for simulated data
- Evaluation possible on downstream classification tasks where features extracted using unsupervised learning methods are used as input features

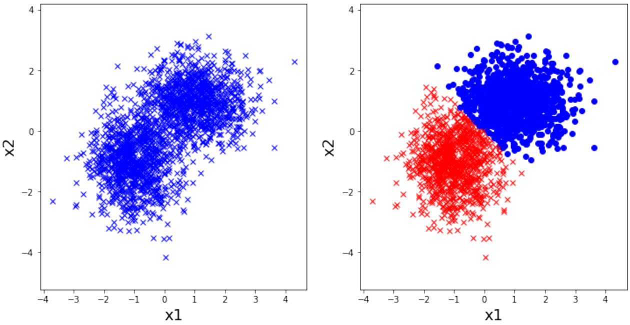
Unsupervised learning in Finance

Why unsupervised learning in finance

- Large amount of unlabelled data is available in most companies
- Data labelling is very costly
- Some applications
  - Cluster customers/assets/portfolio/trend/momentum
  - Infer data generating distribution for stock prices/returns simulation for back-testing
  - Reduce data dimensionality for visualizing high dimensionality data or developping models

Clustering problem

Given a dataset, categorize sample into a pecified number of groups K based on a dissimilarity measure ρ on the sample space.



Samples to the left are clustered into $K = 2$ groups with K-means algorithm based on the the Eucidean distance.

Clustering with K-means

Principle

The goal of the K-means algorithm is to partition a sample set $\{\mathbf{x}_n, n = 1, ..., N\}$ into K sub-sets with assignment variable $s_n, n = 1, ..., N$ minimizing the intra-class distance :

$$\hat{\mathbf{S}} = \arg \min_{s_1,\ldots,s_N} \sum_{n=1}^{N} ||\mathbf{x}_n - \mu_{s_n}||_2$$

where $\mu_k = \frac{1}{|S_k|} \sum_{n \in S_k} \mathbf{x}_n$ with $S_k = \{n | s_n = k\}$

Algorithm

1. Initialize K cluster centroids μ_k : select K samples among the dataset
2. Select a random sample $\mathbf{x}_n$, assign it to the closest centroid's cluster
3. Update the cluster centroid using the new sample $\mathbf{x}_n$
4. loop back to step 2 until convergence

K-means clustering parameters

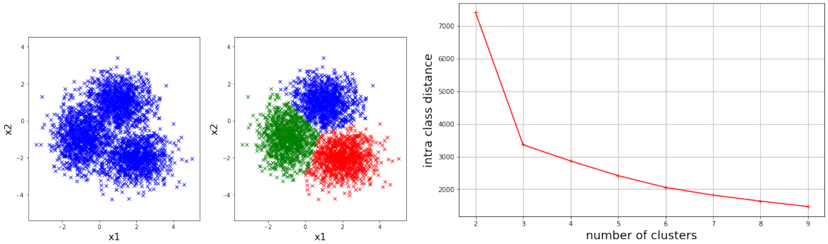
Algorithm requirements

- The number of clusters K
- The dissimilarity measure $\rho(\mathbf{x}_n, \mu_k) = ||\mathbf{x}_n - \mu_k||_2$
- These parameters are specified by the data scientist : you

Specifying the number of cluster K : elbow method

Principle

Run k-mean clustering for varying $K = 1, 2, ..., 10$ and detect inflexion number in the intra class distance curve.



Distance metric definition for K-means

Distance metric choice

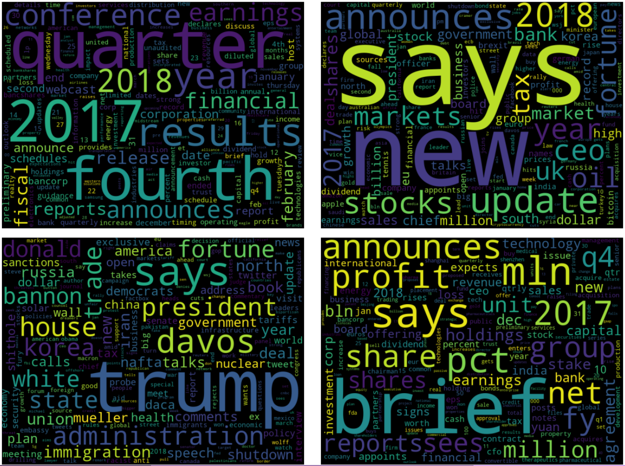
- Classically the Euclidean distance is used with K-means :

$$||\mathbf{x}_n - \mu_k||_2 = \sqrt{\sum_{d=1}^{D}(x_{nd} - \mu_{kd})^2}$$

- Euclidean distance is only suited for vector space elements $\mathbf{x}_n \in \mathbb{R}^D$
- Example : stock return distributions clustering
  - Euclidean distance is not suited to compare distributions
- Important : cluster centroids computation is distance-dependant

K-means example : Finance news corpus clustering

- Build vocabulary : set of wof appearing in documents
- Represent each document as the vector of word occurences
- Apply K-means to word occurence vectors



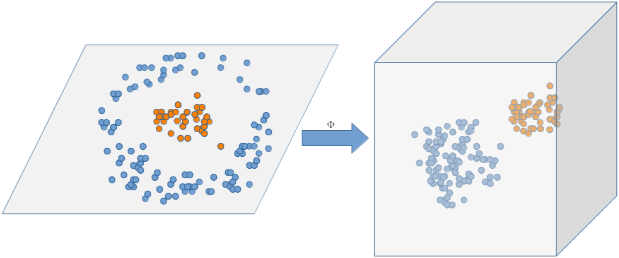
K-means limitations

- Question : why K-means will have trouble on the following problems ?
  - Answer : cluster boundaries are nonlinear
  - Solution : Kernelized K-means

Kernel method for clustering

Principle

Map samples $\mathbf{x}_n$ in a feature space using a map $\phi(\mathbf{x}_n)$ such that in the feature space clustering can be easily solved.



Kernel trick

Exploit the existence of a kernel κ such that $\kappa(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') >$.

Kernel K-means

Principle

The goal of the Kernel K-means is to partition a sample set $\{\mathbf{x}_n, n = 1, ..., N\}$ into K sub-sets with assignment variables $s_n \in \{1, 2, ..., K\}, n = 1, ..., N$ minimizing the intra-class distance :

$$\text{argmin}_{s_n \in \{1,2,...,K\}} \sum_{n=1}^{N} ||\phi(\mathbf{x}_n) - \mu_{s_n}||_2^2$$

such that $\mu_k = \frac{1}{|S_k|} \sum_{s_n=k} \phi(\mathbf{x}_n)$

Issue

- It is not always possible to compute $\mu_k = \frac{1}{|S_k|} \sum_{s_n=k} \phi(\mathbf{x}_n)$ as $\phi(\mathbf{x}_n)$ can be in some cases an infinite dimension vector
- Exploit kernel trick $\kappa(\mathbf{x}, \mathbf{x}') = <\phi(\mathbf{x}), \phi(\mathbf{x}')>$

Kernel K-means objective with kernel trick

Given relations

- Centroid formula : $\mu_k = \frac{1}{|S_k|} \sum_{s_n=k} \phi(\mathbf{x}_n)$
- Inner product relation $||u - v||_2^2 = ||u||_2^2 - 2 < u, v > + ||v||_2^2$
- Kernel trick : $\kappa(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') >$

Objective function

$$||\phi(\mathbf{x}_n) - \mu_{s_n}||_2^2 = \kappa(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{|S_{s_n}|} \sum_{m \in S_{s_n}} \kappa(\mathbf{x}_n, \mathbf{x}_m) + \frac{1}{|S_{s_n}|^2} \sum_{m, m' \in S_{s_n}} \kappa(\mathbf{x}_m, \mathbf{x}_{m'})$$

Greedy kernel K-means algorithm

- Choose a number of cluster K, and a kernel $\kappa(\mathbf{x}_n, \mathbf{x}_m)$
- Randomly initialized sample cluster assignments $s_n, n = 1, ..., N$
- Repeat until convergence
  1. Select a sample $\mathbf{x}_n$
  2. For $k = 1, 2, ..., K$ compute

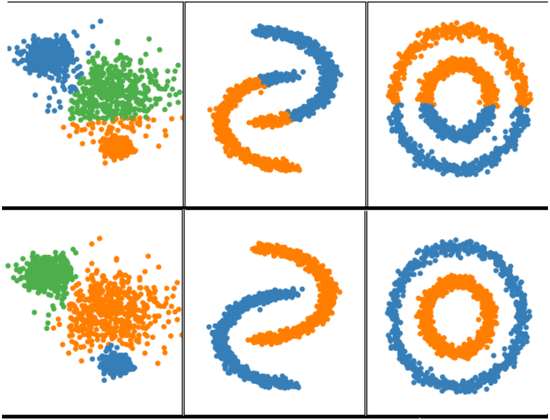
$$L(k) = \kappa(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{|S_k|} \sum_{m \in S_k} \kappa(\mathbf{x}_n, \mathbf{x}_m) + \frac{1}{|S_k|^2} \sum_{m, m' \in S_k} \kappa(\mathbf{x}_m, \mathbf{x}_{m'})$$

  3. Assign sample $\mathbf{x}_n$ to cluster minimizing $L(k)$:

$$s_n = \arg \min_{k=1,\ldots,K} L(k)$$

K-means vs Kernel K-means

- First row : K-means clustering
- Second row : radial basis function kernel K-means clustering
- Kernel K-means is more efficient at clustering non-spherical samples



Overview

1. Reminders about previous sessions
2. Unsupervised learning
   - Clustering
     - Clustering with K-means
     - Kernel K-means
   - Density estimation
     - Non parametric kernel densities
     - Mixture model densities
   - Dimensionality reduction
     - Principal component analysis (PCA)
     - Kernel Principal component analysis (PCA)
     - Neural embeddings

Density representation

What is a density

- Density represents sample probability distributions : Gaussian density
- Machine learning is interested in complex/multi modal densities
  - Non parametric densities : kernel based densities
  - Mixture models : Gaussian mixture models

Applications

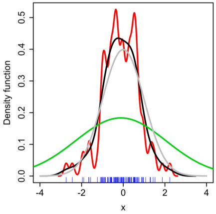
- Outliers detection : finding low probability samples
- Simulations : draw samples from densities

Non parametric kernel densities

Assume $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ identically distributed sampled from an unknow density f, at any given point $f(\mathbf{x})$ can be estimated as :

$$\hat{f}(\mathbf{x}) = \frac{1}{C(h, N)} \sum_{n=1}^{N} K_h(\frac{\mathbf{x} - \mathbf{x}_n}{h})$$

- K_h : kernel (as for kernel methods)
- h kernel bandwidth
- $C(h, N)$: normalization factor



Parametric density estimation with E.M. algorithm

Expectation Maximization (E.M.)

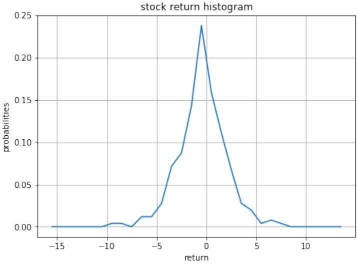
- Data is represented as a mixture of a known distribution (e.g. Gaussian, multinomial, Dirichlet)

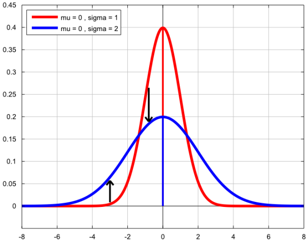
$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k p_k(\mathbf{x})$$

- E.M. produces a density distribution of the given dataset
- For each sample, E.M. produces an assignment score to each mixture
- E.M. is a generalization of k-means algorithm
- **Application** : stock returns modelling beyond Gaussians

Non Gaussian stock return histogram example

- Non Gaussian stock return distribution : too skewed
- Can be modelled by a mixture of two gaussians





Unsupervised learning of a G.M.M. with E.M.

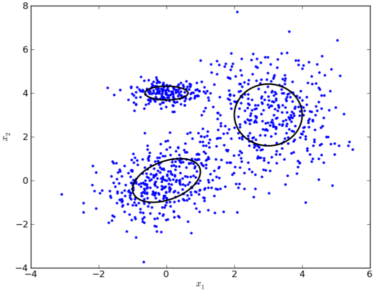
D-dimensional Gaussian density

$$g(\mathbf{x}_n; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} exp - \frac{1}{2}(\mathbf{x}_n - \mu_k)^t \Sigma_k^{-1} (\mathbf{x}_n - \mu_k)$$

Problem

Given a dataset $(\mathbf{x}_n)$, learn using EM a gaussian mixture model (GMM)

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} \alpha_k g(\mathbf{x}_n; \mu_k, \Sigma_k)$$



Learning a GMM with EM algorithm : principle

Latent variables

Introduce assignment variable Z_n such that $p(Z_n = k) = \alpha_k$ is the prior probability sample $\mathbf{x}_n$ being generated by the mixture component k.

Maximum marginal likelihood

Estimate the GMM parameters $\Theta = (\alpha_k, \mu_k, \Sigma_k)$ maximizing likelihood :

$$
\begin{aligned}
p(\mathbf{x}_{1:N}, Z_{1:N}|\Theta) &= \prod_{n=1}^{N} p(\mathbf{x}_n|Z_n)p(Z_n) \\
&= \prod_{n=1}^{N}\prod_{k=1}^{K}[p(\mathbf{x}_n|Z_n = k)p(Z_n = k)]^{\delta_{Z_n}(k)} \\
&= \prod_{n=1}^{N}\prod_{k=1}^{K}[g(\mathbf{x}_n; \mu_k, \Sigma_k)\alpha_k]^{\delta_{Z_n}(k)}
\end{aligned}
$$

Learning a GMM with EM algorithm : the algorithm

Expectation step :

Compute expectation of log-marginal likelihood wrt observations :

$$Q(\Theta|\Theta_{t-1}) = \mathbb{E}_{Z_{1:N}|\mathbf{X}_{1:N},\Theta_{t-1}}[\log p(\mathbf{X}_{1:N}, Z_{1:N}|\Theta)]$$

Maximization step :

Maximize expectation wrt to parameters :

$$\Theta_t = \arg\max_{\Theta} Q(\Theta|\Theta_{t-1})$$

EM Algorithm :

1. Initialize the GMM parameters Θ_0, set $t = 1$
2. Compute log-marginal likelihood expectation $Q(\Theta|\Theta_{t-1})$
3. Compute Θ_t as the arg-maximum of $Q(\Theta|\Theta_{t-1})$
4. Increment t, and loop back to set 2 until convergence

Learning the number of mixtures

As for K-means elbow method on the number of mixture vs the log likelihood curves can be used to estimate the optimal number of mixtures.

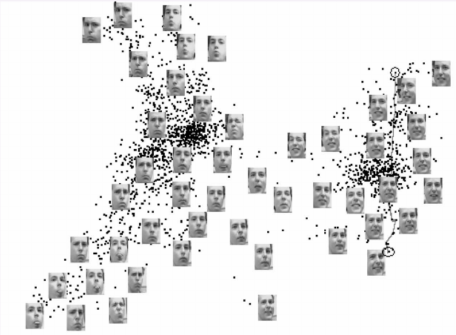
Overview

1. Reminders about previous sessions
2. Unsupervised learning
    - Clustering
        - Clustering with K-means
        - Kernel K-means
    - Density estimation
        - Non parametric kernel densities
        - Mixture model densities
    - Dimensionality reduction
        - Principal component analysis (PCA)
        - Kernel Principal component analysis (PCA)
        - Neural embeddings

What and why dimensionality

What

- Finding a lower dimensional representation of samples

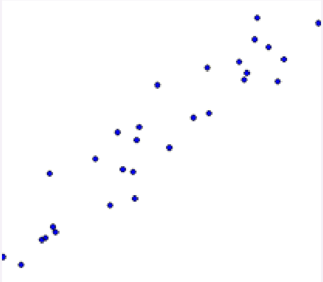


Why

- Additional dimension can je just ud to noise in data
- Models can be easier to build on lower dimensional

Problem statement

Principal component analysis (PCA) can be defined as the orthogonal projection of the data on a lower dimensional linear space such that the variance of the projected data is maximized.



Dimensionality reduction with PCA : principle (1/2)

- Consider samples $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$ with $\mathbf{x}_n \in \mathbb{R}^D$
- Goal : project samples on M-dimension linear space ($M < D$) while maximizing projected sample variance

Case $M = 1$

- Principal sub-space is defined vector $\mathbf{u}_1$ with $||\mathbf{u}_1||_2^2 = \sqrt{\mathbf{u}_1^\top \mathbf{u}_1} = 1$
- Covariance matrix of a centered sample set ($\bar{\mathbf{x}} = 0$) is defined as :

$$S = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top$$

- Projected samples on subspace spanned by $\mathbf{u}_1$ are given by $\tilde{x}_n = \mathbf{u}_1^\top \mathbf{x}_n$
- Projected samples variance is defined as :

$$\frac{1}{N} \sum_{n=1}^{N} (\mathbf{u}_1^\top \mathbf{x}_n)^2 = \mathbf{u}_1^\top S \mathbf{u}_1$$

Dimensionality reduction with PCA : principle (2/2)

- Problem : find vector $\mathbf{u}_1$ maximizing $\mathbf{u}_1^\top S \mathbf{u}_1$ with $\mathbf{u}_1^\top \mathbf{u}_1 = 1$
- Introducing Langrange multiplier λ_1 optimization problem is :

$$J(\mathbf{u}_1) = \mathbf{u}_1^\top S \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$$

- Computing gradient of $J(\mathbf{u}_1)$ wrt $\mathbf{u}_1$ and setting to 0

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

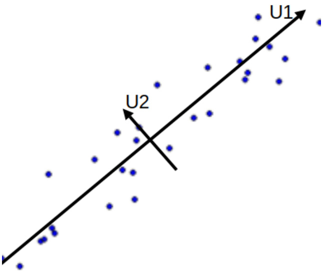
Solution

- $\mathbf{u}_1$ is an eigenvector of covariance matrix S with eigenvalue λ_1
- $\mathbf{u}_1^\top S \mathbf{u}_1 = \lambda_1$: is maximized if λ_1 is the largest eigenvalue, and $\mathbf{u}_1$ its eigenvector

$M > 1$

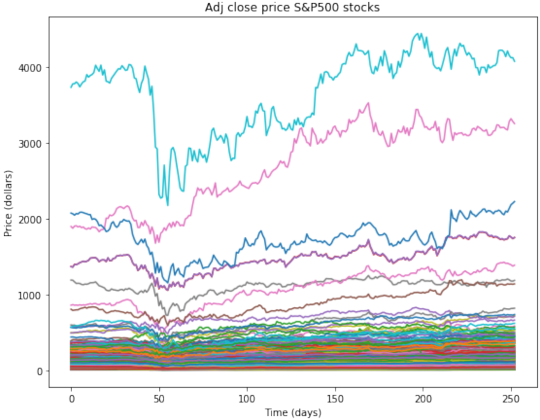
Select the M eigen vectors corresponding to the largest eigenvalues

PCA : two dimensions example

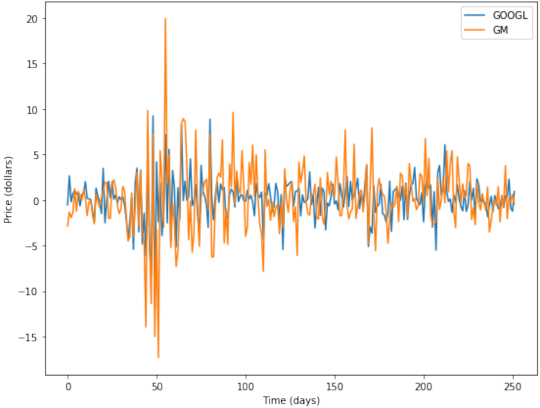


- Two principal linear subspaces $\mathbf{u}_1$ and $\mathbf{u}_2$
  - $\mathbf{u}_1$: first principal sub-space corresponding to largest eigenvalues
  - $\mathbf{u}_2$: second principal sub-space
- Data can be considered unidimensional according to first eigenspace

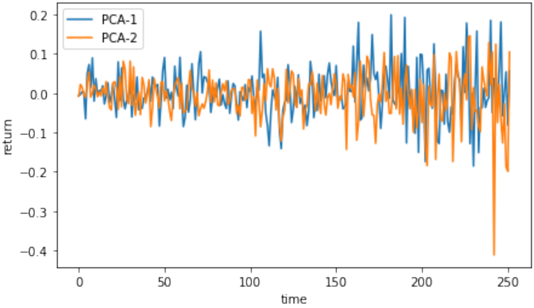
Stock return trajectories PCA analysis : stock prices



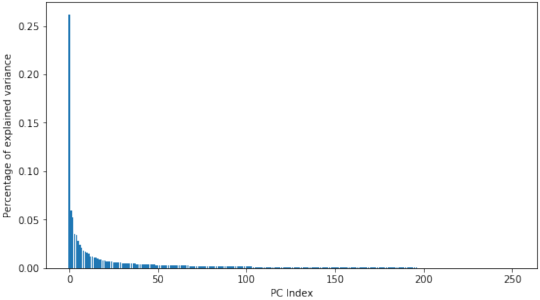
$$\text{return}_t = \frac{\text{price}_t - \text{price}_{t-1}}{\text{price}_{t-1}}$$



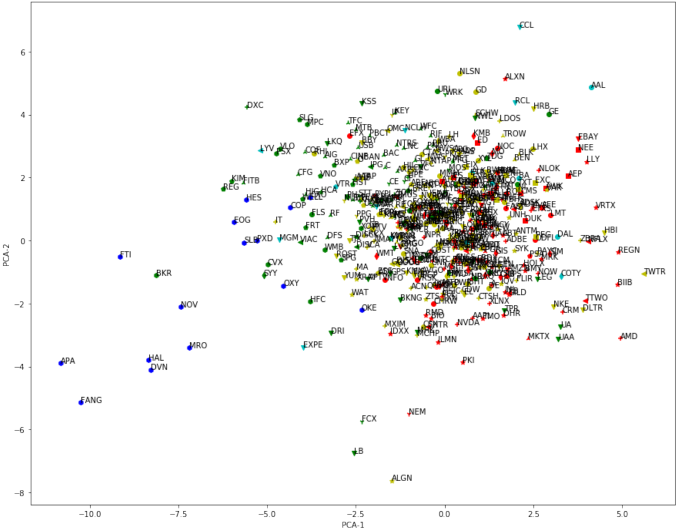
Stock return trajectories PCA analysis : two first PCs



Stock return trajectories PCA analysis : explained variance



Stock return trajectories PCA analysis : embedding



PCA applied to stock returns : eigen portfolios

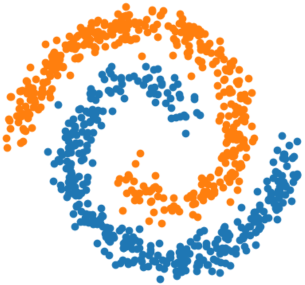
- Compute PCA stock returns decomposition
- First PC corresponds to market trend
- Portfolio with weights corresponding coordinate wrt the first PC

$$w_k \propto < \mathrm{PC}_1, \mathrm{S}_k >$$

is strongly correlated to S&P index

Issues with linear principal component analysis

- Favorable case : samples have a linear span.



- Unfavorable case : samples have a curvilinear span.
- Solutions :
  - Kernel PCA
  - Neural embedding

Kernel PCA : principles

Feature map and associated kernel

Let $\phi : \mathbb{R} \longleftrightarrow \mathcal{H}$ be a feature map and κ its associated kernel :

$$\kappa(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') >_{\mathcal{H}}$$

Principles

Given dataset $\mathbf{x}_n, n = 1, ..., N$:

- Map samples in a feature space using feature map ϕ
- Apply PCA to the mapped samples $\phi(\mathbf{x}_n), n = 1, ..., N$
- Exploit kernel trick $\kappa(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') >$ to simplify computations

Kernel PCA : the optimization problem

Orthogonal projection in feature space

Let f be a direction in the feature space, the orthogonal projection of mapped feature on direction f is given by :

$$h_f(\mathbf{x}) = < \phi(\mathbf{x}), \frac{f}{||f||} >$$

Orthogonal projection's variance is given by :

$$\text{var}[h_f] = \frac{1}{N} \sum_{n=1}^{N} \frac{< \phi(\mathbf{x}_n), f >^2}{||f||^2}$$

Principal components in feature space are solutions of the problem :

$$f_i = \begin{cases} \arg\max_{f \perp \{f_1, \ldots, f_{i-1}} \text{var}[h_f] \\ \text{with } ||f|| = 1 \end{cases}$$

Kernel PCA : eigenvalue problem (1/2)

The representer theorem

for all $\mathbf{x}$ there exist $\alpha_i = (\alpha_{i1}, ..., \alpha_{iN})$ such that $f_i(\mathbf{x}) = \sum_{n=1}^{N} \alpha_{in}\kappa(\mathbf{x}_n, \mathbf{x})$

$$||f_i||^2 = \sum_{n,m}^{N} \alpha_{in}\alpha_{im}\kappa(\mathbf{x}_n, \mathbf{x}_m) = \alpha_i^\top K \alpha_i$$

$$\sum_{n=1}^{N} f_i(\mathbf{x}_n)^2 = \alpha_i^\top K^2 \alpha_i$$

$$< f_i, f_j >= \alpha_i^\top K \alpha_j$$

$$\alpha_i = \begin{cases} arg\max_{\alpha \in \mathbf{R}^N} \alpha^\top K \alpha \\ \alpha_i^\top K \alpha_j = 0, j = 1, ..., i-1 \\ \alpha_i^\top K \alpha_i = 1 \end{cases}$$

Kernel PCA : eigenvalue problem (2/2)

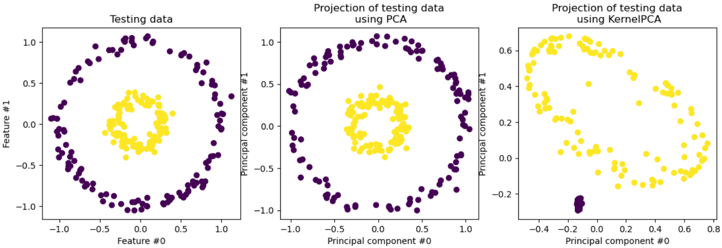
- Decomposing $K = U\Delta U^\top$ with eigenvalues $\Delta_1, ..., \Delta_N \geq 0$
- setting $\beta = K^{\frac{1}{2}}\alpha$ with $K^{\frac{1}{2}} = U\Delta^{\frac{1}{2}}U^\top$

Eigenvalues problem

$$\beta_i = \begin{cases} \mathit{arg}\,\max_{\beta\in\mathbf{R}^N} \beta^\top K\beta \\ \beta_i^\top\beta_j = 0, j = 1, ..., i-1 \\ \beta_i^\top\beta_i = 1 \end{cases}$$

Kernel PCA : scikit learn example

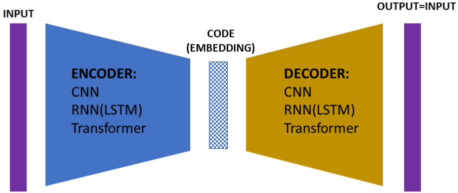
- Model comparison over a non-spherical dataset
- Projection on two first principal components
- Kernel PCA efficiently identifies the two curves of variation

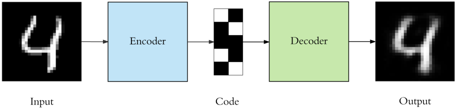


Neural embeddings

Auto-encoders

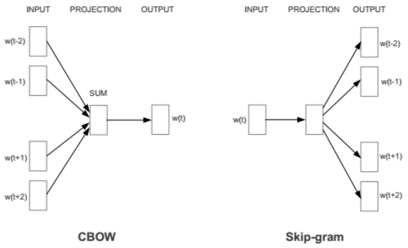
- Networks trained to replicate their inputs through a bottleneck layer
- Bottleneck forces encoder to compress input into low dimension code
- Decoder reconstructs input from low dimension code
- Code contain all information required to reconstruct into inputs



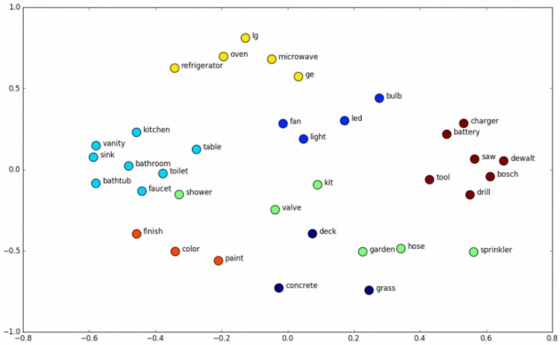


Definition

- A word embeddding model is trained to predict a word through a bottleneck (embedding) using the word context (or vice versa)
- Allows to go beyond linear PCA dimensionality reduction



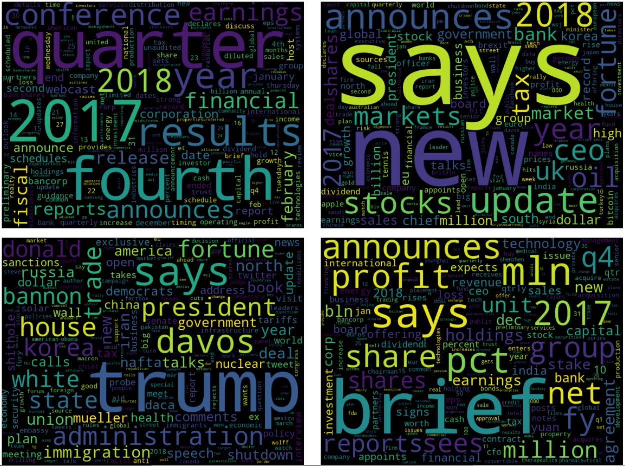
Word embedding results



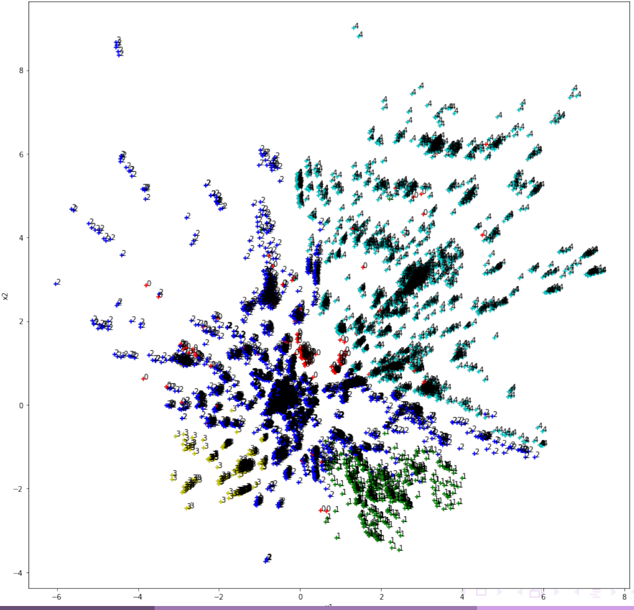
In the embedding space, semantically related words are close

Finance news corpus document embedding : clustering

- Build vocabulary : set of wof appearing in documents
- Represent each document as the vector of word occurences
- Apply K-means to word occurence vectors

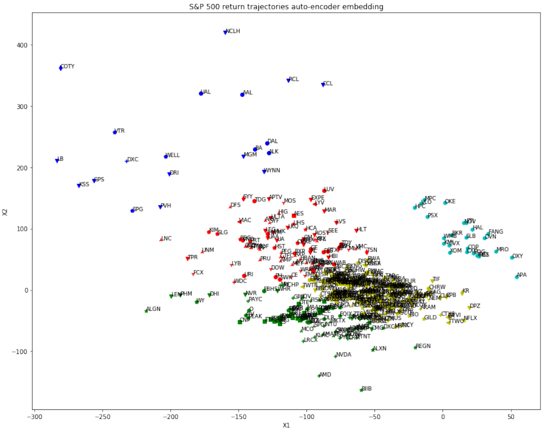


Finance news corpus document embedding



Stock return trajectories neural embedding (1/2)

- An auto-encoder is train to replicated stock returns trajectories through a 2 dimension bottleneck
- Initial trajectories is compressed into 2 dimensions



Stock return trajectories neural embedding (2/2)

- Stocks returns in plot according to mean return vs volatility
- Neural embedding captured well mean return volatility structure