PCA and Kernel PCA basics

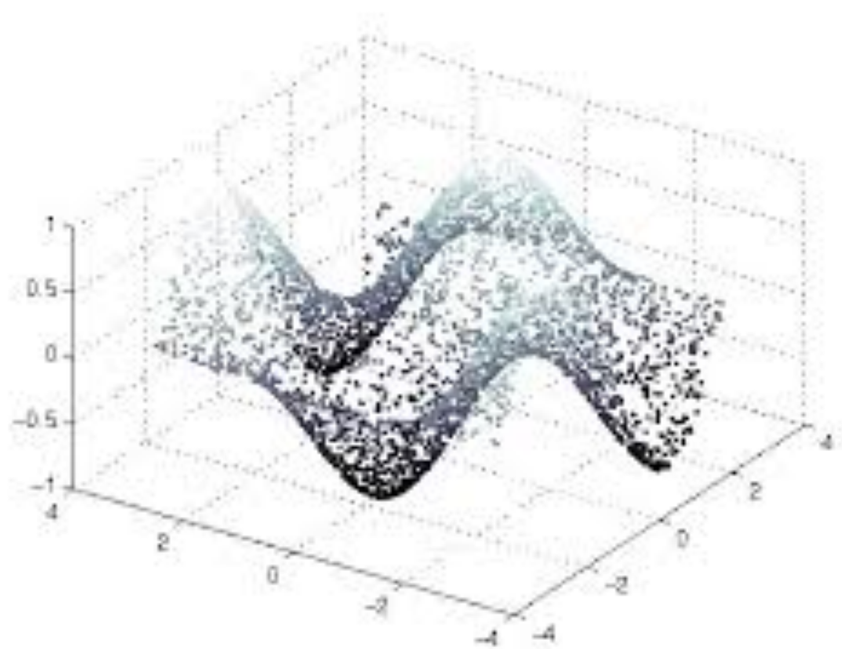
Olivier J.J. MICHEL, Florent CHATELAINn

GIPSA-Lab, UMR 5216 CNRS

Continuous Latent variables , PCA

Motivations :

- Curse of dimensionality: it is not possible to get enough data to cover all the observation space. High dimensional spaces are mostly empty !
- Multivariate data live mostly in a (unknown) lower dimensional space : Intrinsic Dimensionality << representation dimensionality
- Noise "visits" all possible dimensions : dimension reduction $\simeq$ noise reduction

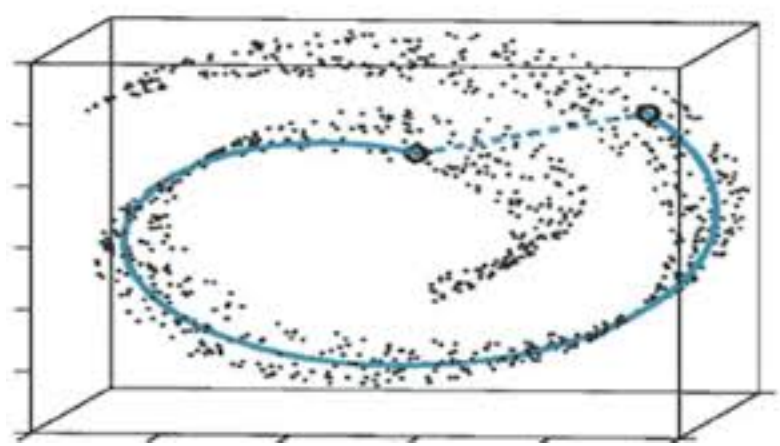


Motivations (cont'd) :

- features extraction (+ reduction of redundancy)
- Lossy compression (reduction of the size of the data)
- data visualization

Extraction techniques (manifold learning)

- Physically based methods, geodesic distances
- Statistical methods, dictionary learning
- filtering (linear/non linear)



Classical (non probabilistic) PCA

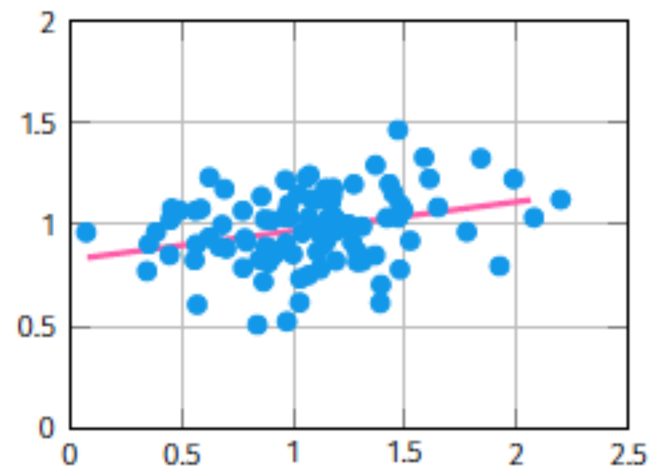
Simplest approach : assumes

- Euclidean subspace
- Gaussian dist. for latent variables
- linear Gaussian dependancies of observed variables and state of latent var.

- Find a linear transform to reduce the dimensionality of the data

$$z_i = < \mathbf{v}_i, \mathbf{x} >$$

- Find ('new') low dimensional feature vector $\mathbf{z}$ that account most of the variability of the data:
  - $z_1, z_2, z_3, \ldots$ are jointly uncorrelated
  - $\mathrm{var}(z_i)$ are as large as possible
  - $\mathrm{var}(z_1) \geq \mathrm{var}(z_2) \geq \mathrm{var}(z_3) \geq \ldots$



Computing PCA

- Find $\mathbf{v}_1$ s.t. var(z_1) is max, and constraint $||\mathbf{v}_1||^2 = 1$

$$\text{var}(z_1) = \text{var}(< \mathbf{v}_1, \mathbf{x} >) = \mathbf{v}_1^T \Gamma \mathbf{v}_1$$

- Form the Lagrangian

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^T \Gamma \mathbf{v}_1 + \lambda_1 (1 - \mathbf{v}_1^T \mathbf{v}_1)$$

- Compute the gradient wrt $\mathbf{v}_1$

$$\nabla_{\mathbf{v}_1} \mathcal{L} = 2\Gamma \mathbf{v}_1 - 2\lambda \mathbf{v}_1$$

- $\mathbf{v}_1$ is an eigenvector of the cov. matrix of $\mathbf{x}$:

$$[\nabla_{\mathbf{v}_1} \mathcal{L} = 0] \Leftrightarrow [\Gamma \mathbf{v}_1 = \lambda \mathbf{v}_1]$$

- Maximize var(z_1) :

$$\text{var}(z_1) = \mathbf{v}_1^T \Gamma \mathbf{v}_1 = \lambda_1 \mathbf{v}_1^T \mathbf{v}_1 = \lambda_1$$

- Find $\mathbf{v}_2$ s.t. $\mathrm{var}(z_2)$ is max, but $||\mathbf{v}_2||^2 = 1$ and $< \mathbf{v}_1, \mathbf{v}_2 >= 0$

- Form the Lagrangian

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_{12}) = \mathbf{v}_2^T \Gamma \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^T \mathbf{v}_2) + \beta_{12}(0 - \mathbf{v}_1^T \mathbf{v}_2$$

- Compute the gradient wrt $\mathbf{v}_2$

$$\nabla_{\mathbf{v}_2} \mathcal{L} = 2\Gamma \mathbf{v}_2 - 2\lambda_2 \mathbf{v}_2 - \beta_{12} \mathbf{v}_1$$

- Solve $\nabla_{\mathbf{v}_1} \mathcal{L} = 0$

- Use orthogonality constraint $< \mathbf{v}_1, \mathbf{v}_2 >= 0$, multiplying the equatio
below by $\mathbf{v}_1^T$:

$$\begin{aligned}
\Gamma \mathbf{v}_2 &= \lambda_2 \mathbf{v}_2 + \frac{\beta_{12}}{2} \mathbf{v}_1 \\
\mathbf{v}_1^T \Gamma \mathbf{v}_2 &= \frac{\beta_{12}}{2} \\
\lambda_1 \mathbf{v}_1^T \mathbf{v}_2 &= \frac{\beta_{12}}{2} \\
0 &= \beta_{12}
\end{aligned}$$

- Finaly: $\Gamma \mathbf{v}_2 = \lambda_2 \mathbf{v}_2 \Rightarrow \mathbf{v}_2$ is the 2^{nd} largest eigenvalue of Γ

Computing PCA in practice

1. Estimate (ML) the mean of $\mathbf{x}$ ($\in \mathbb{R}^d$)

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

2. Estimate (ML) the covariance matrix :

$$\Gamma = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$

3. Compute the p largest eigenvalues and eigenvectors of Γ, and select $p < d$ wrt "explained variance" :

$$\frac{\sum_{k=1}^{p} \lambda_k}{\sum_{k=1}^{d} \lambda_k}$$

!Note! : STANDARDIZATION or SCALING MATTER

RK : Step 3 may be difficult / costly

See PCA_IRIS_example.ipynb

PCA for high dimensional data (N<d)

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}_1 - \mu)^T \\ (\mathbf{x}_2 - \mu)^T \\ \vdots \\ (\mathbf{x}_N - \mu)^T \end{bmatrix} \in \mathbb{R}^N \times \mathbb{R}^d \Rightarrow \Gamma = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

Eigendecomposition équation ($\Gamma \in \mathbb{R}^d \times \mathbb{R}^d$)

$$\begin{aligned} \Gamma \mathbf{v}_i &= \lambda_i \mathbf{v}_i \\ \frac{1}{N-1}\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{v}_i &= \lambda_i \mathbf{X}\mathbf{v}_i \end{aligned}$$

Equivalent to $S\mathbf{u}_i = \lambda_i \mathbf{u}_i$ where $\mathbf{u}_i = \mathbf{X}\mathbf{v}_i$ and $S = \frac{1}{N-1}\mathbf{X}\mathbf{X}^T \in \mathbb{R}^N \times \mathbb{R}^N$:
Eigenproblem in $O(N^2)$ instead of $O(d^2)$

In practice : Compute $\mathbf{u}_i$ then $\mathbf{v}_i$ s.t. $||\mathbf{v}_i|| = 1$

$$\mathbf{v}_i = \frac{1}{\sqrt{(N-1)\lambda_i}}\mathbf{X}^T\mathbf{u}_i$$

Kernel PCA : PCA with the KERNEL Trick!

(as PCA depends only on inner products)

Let k be a positive semi-definitive kernel function

$$\begin{aligned} k : \mathbb{R}^d \times \mathbb{R}^d &\rightarrow \mathbb{R} \\ (\mathbf{x}, \mathbf{x}') &\mapsto k(\mathbf{x}, \mathbf{x}') \equiv < \Phi(\mathbf{x}), \Phi(\mathbf{x}') > \end{aligned}$$

$$\Gamma = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T \leftarrow C = \frac{1}{N} \sum_{n=1}^{N} \Phi(\mathbf{x}_n), \Phi(\mathbf{x}_n)^T$$

New eigenproblem :

$$C\mathbf{v}_i = \lambda_i \mathbf{v}_i \Leftrightarrow \frac{1}{N} \sum_n \Phi(\mathbf{x}_n)[\Phi(\mathbf{x}_n)^T \mathbf{v}_i] = \lambda_i \mathbf{v}_i$$

but $\mathbf{v}_i = \sum_m a_{im} \Phi(\mathbf{x}_n)$ thus the eigenequation becomes

$$\left(\frac{1}{N} \sum_{n=1}^{N} \Phi(\mathbf{x}_n) \Phi(\mathbf{x}_n)^T \right) \left(\sum_m a_{im} \Phi(\mathbf{x}_n) \right) = \lambda_i \sum_m a_{im} \Phi(\mathbf{x}_n)$$

Multiplying both rhs and lhs terms in the preceding equality. by $\Phi(\mathbf{x}_l)$:

$$\frac{1}{N}\sum_n k(\mathbf{x}_l,\mathbf{x}_n)^T \sum_m a_{im} k(\mathbf{x}_n,\mathbf{x}_m) = \lambda_i \sum_m a_{im} k(\mathbf{x}_l,\mathbf{x}_m)$$

or in matrix form:

$$\begin{aligned} & K^2\mathbf{a}_i = \lambda_i N K \mathbf{a}_i \\ \Rightarrow \quad & K\mathbf{a}_i = \lambda_i N \mathbf{a}_i \end{aligned}$$

In practice :

- normalization

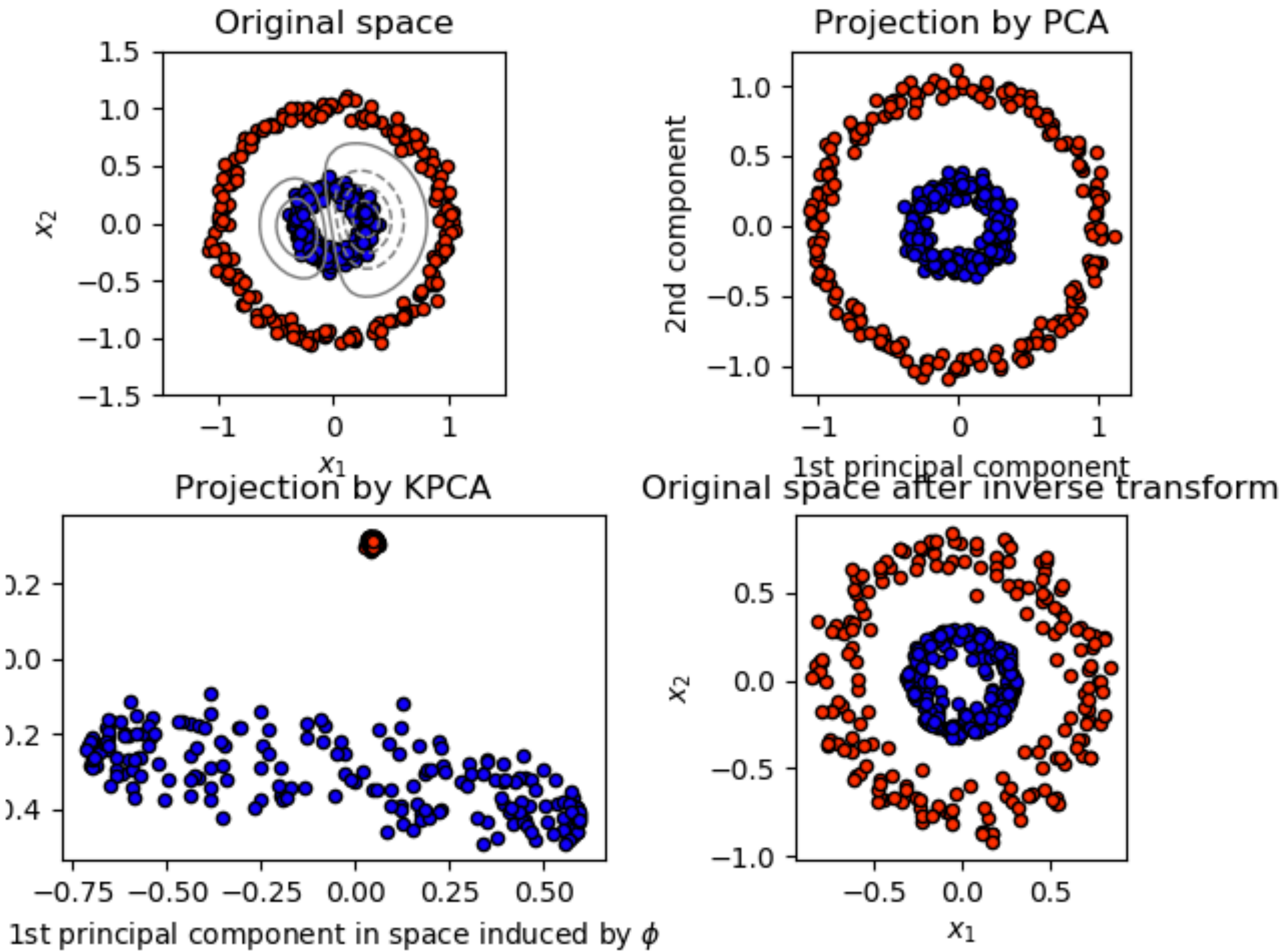
$$1 = \mathbf{v}_i^T\mathbf{v}_i = \mathbf{a}_i^T K \mathbf{a}_i = \lambda_i N \mathbf{a}_i^T \mathbf{a}_i$$

- Projection on eigenvector

$$\mathbf{y}_i(\mathbf{x}_i) = \phi(\mathbf{x}_i) = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{a}\mathbf{x}_n)$$

Example of Kernel PCA with RBF (from Scikit-learn documentation)

This example shows that Kernel PCA is able to find a projection of the data that makes data linearly separable.



See PCA_IRIS_example.ipynb

See PCA_sonardata_example.ipynb