# Representation learning: principal component analysis

Formation ENSTA-ParisTech
Conférence IA

Florent Chatelain [*] Olivier Michel [*]

[*] Univ. Grenoble Alpes, GIPSA-lab

10-13 February 2020

# Outline

## Unsupervised learning dataset

- We observe only the feature vectors $X_i \in \mathbb{R}^p$, for $i = 1, \ldots, n$
- There is no associated targets/responses $Y_i$

## Objectives

Different problems than prediction (no response $Y$ to predict):

- find subgroups of features that behave similarly
- ☞ *clustering*, broad class of methods for discovering unknown subgroups in data
- discover some insights or relations among the data/observations
- ☞ *principal component analysis (PCA):* find informative way to represent/visualize data

## Illustration

- Curse of dimensionality: it is not possible to get enough data to cover all the observation space.

  *High dimensional spaces are mostly empty!*

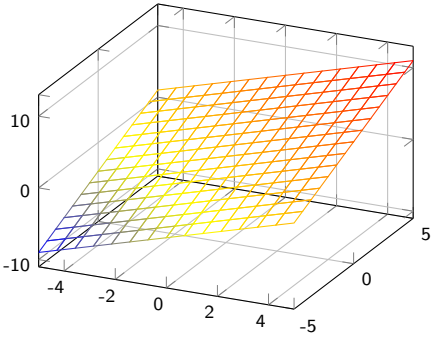
- Informative data usually live in a lower dimensional space, but which one?

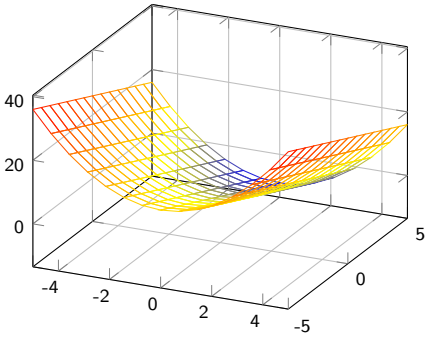
## Illustration

- Curse of dimensionality: it is not possible to get enough data to cover all the observation space.

  *High dimensional spaces are mostly empty!*

- Informative data usually live in a lower dimensional space, but which one?



# The Challenge of Unsupervised Learning

## Unsupervised Learning

- Unsupervised learning is a more subjective and difficult problem than supervised learning: there is no simple goal such as prediction of a response
- But easier to get *unlabeled data*, as labeling can require human expertise and intervention

☞ tools as PCA are also useful for data visualization or data pre-processing before to apply supervised techniques

## PCA Motivations

- Multivariate data live mostly in a (unknown) lower dimensional space: Intrinsic Dimensionality $\ll$ representation dimensionality
- Noise "visits" all possible dimensions: dimension reduction $\simeq$ noise reduction

## Application

Representation learning, such as PCA, is of growning interest in a large number of fields

- It reduces the size of the data,
- It limits the redundancy,
- It permits visualization of the data,
- It mitigates the *curse of dimensionality*.

Representation learning, aka *feature learning*, techniques:

- Physically based method,
- filtering (linear/non-linear)
- Statistical methods, e.g. PCA/Dictionary Learning

# Outline

## Goal of Principal Component Analysis

Find a low dimensional space to represent a dataset

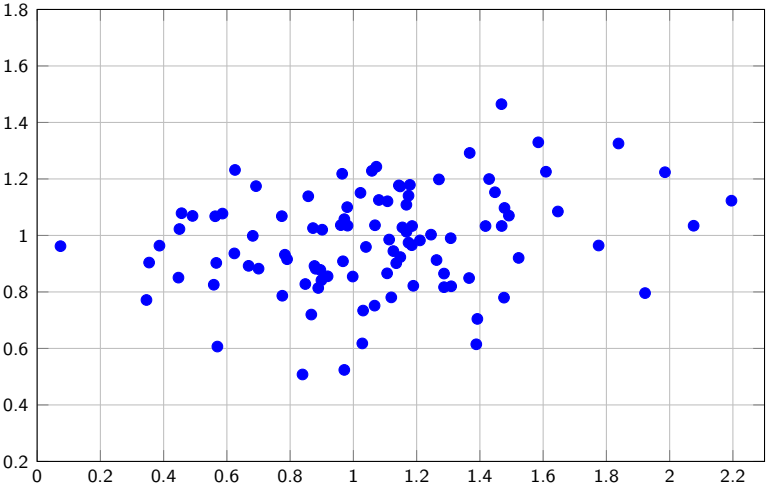
- produce a sequence of linear combination of the variables (feature vector components) with maximal variance (thus that best explains the dispersion of data) and which are mutually uncorrelated

### Linear transformation

new basis $\phi_1,\ \phi_2,\ \phi_3, \ldots$ in $\mathbb{R}^p$ where feature projections $Z_i = \langle \phi_i, X \rangle$ account for most of the variability of the data

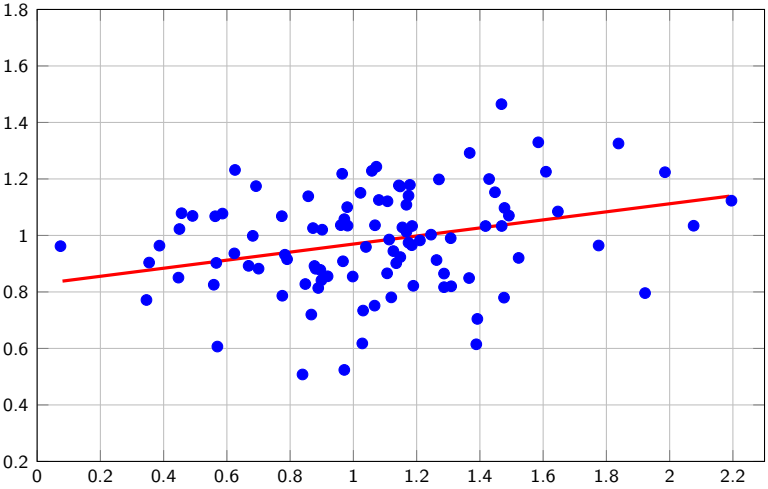
- $Z_1,\ Z_2,\ Z_3, \ldots$ are mutually uncorrelated
- $\text{var}(Z_i)$ is as large as possible,
- $\text{var}(Z_1) > \text{var}(Z_2) > \text{var}(Z_3) > \ldots$
- the *loading* vectors $\phi_1,\ \phi_2,\ \phi_3, \ldots$ are the principal component directions

## PCA example



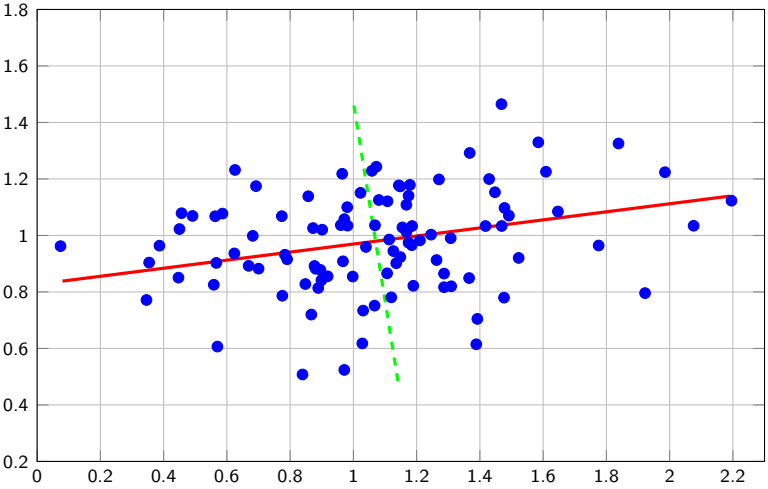

Bivariate data points $X_i = (X_{i1}, X_{i2}) \in \mathbb{R}^2$, $i = 1, \ldots, 100$. The red solid line indicates the first principal component direction, and the green dashed line indicates the second principal component direction.

## PCA example




Bivariate data points $X_i = (X_{i1}, X_{i2}) \in \mathbb{R}^2$, $i = 1, \ldots, 100$. The red solid line indicates the first principal component direction, and the green dashed line indicates the second principal component direction.

## PCA example




Bivariate data points $X_i = (X_{i1}, X_{i2}) \in \mathbb{R}^2$, $i = 1, \ldots, 100$. The red solid line indicates the first principal component direction, and the green dashed line indicates the second principal component direction.

## Details of Principal Component Analysis

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be the data set ($n$ samples of $p$-dimensional vectors)

- ☞ each column $j = 1, \ldots, p$ is a variable
- ☞ each line $i = 1, \ldots, n$ is a sample (of $p$ variables)

We denote as

- $X_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the $i$th line of $\mathbf{X}$, $i = 1, \ldots, n$
- $\phi_1 = (\phi_{11}, \ldots, \phi_{p1}) \in \mathbb{R}^p$ is the first principal component direction, aka the first loading vector
- $z_{i1} = \langle \phi_1, X_i \rangle = \phi_{11} x_{i1} + \cdots + \phi_{p1} x_{ip}$ is the principal component score of the $i$*th* sample, $i = 1, \ldots, n$.

## Details of Principal Component Analysis (Cont'd)

Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centerd to have mean zero (i.e. the column means of $\mathbf{X}$ are zero)

- the scores $z_{i1} = \phi_{11}x_{i1} + \cdots + \phi_{p1}x_{ip}$ have zero mean, and the sample variance to maximize reduces to

$$\max_{\phi_1} \sum_{i=1}^{n} z_{i1}^2 = ||\mathbf{X}\phi_1||^2$$

- we constraint the loadings so that $\phi_1$ is a unit vector: $||\phi_1||^2 = \phi_{11}^2 + \cdots + \phi_{p1}^2 = 1$ (otherwise setting these elements to an arbitrary large value results in an arbitrary large variance)

## Computation of Principal Component Analysis

Let $\mathbf{S} = \mathbf{X}^T\mathbf{X}$ be the sample covariance matrix. It can be diagonalized in an orthogonal basis: $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where

- $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ is a diagonal matrix with by convention $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$
- $\mathbf{U}$ is an orthogonal matrix: $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}_p$ whose columns are the eigenvectors of $\mathbf{S}$

## Computation of Principal Component Analysis (Cont'd)

The maximum variance criterion expresses as

$$\max_{\phi_1,\ \text{s.t.}||\phi_1||=1} ||\mathbf{X}\phi_1||^2 = \phi_1^T \mathbf{X}^T \mathbf{X} \phi_1 = \phi_1^T \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \phi_1,$$

Let $w_1 = \mathbf{U}^T \phi_1$, then $||w_1|| = ||\phi_1||$ and the criterion becomes

$$\max_{w_1,\ \text{s.t.}||w_1||=1} w_1^T \mathbf{\Lambda} w_1 = \lambda_1 w_{11}^2 + \cdots + \lambda_p w_{p1}^2 \leq \lambda_1,$$

- the first principal component direction $\phi_1$ is the eigenvector of $\mathbf{S} = \mathbf{X}^T\mathbf{X}$ associated to the eigenvalue $\lambda_1$ which is the variance of this component
- the second principal component direction $\phi_2$ is the eigenvector associated to $\lambda_2$ (maximal variance among all linear combination uncorrelated to $\phi_1$)
- ...

# Computation of Principal Component Analysis (Cont'd)

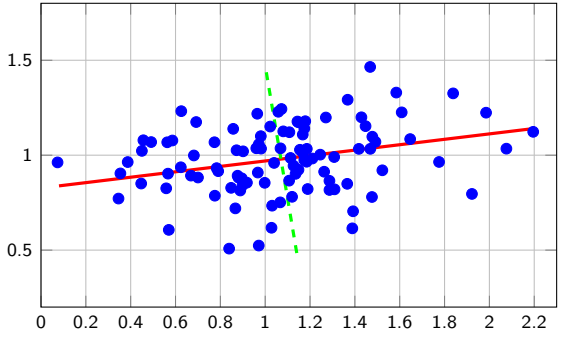
## PCA is an eigenvalue decomposition

The principal component directions $\phi_1, \phi_2, \phi_3, \ldots$ are the ordered sequence of eigenvectors of the sample covariance matrix $\mathbf{S}$, and the variances of the components are the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_2 \geq \ldots \geq 0$

- constraining the $Z_i$ to be mutually uncorrelated is equivalent to constrain the principal component direction $\phi_i$ to be orthogonal

## Geometry of PCA

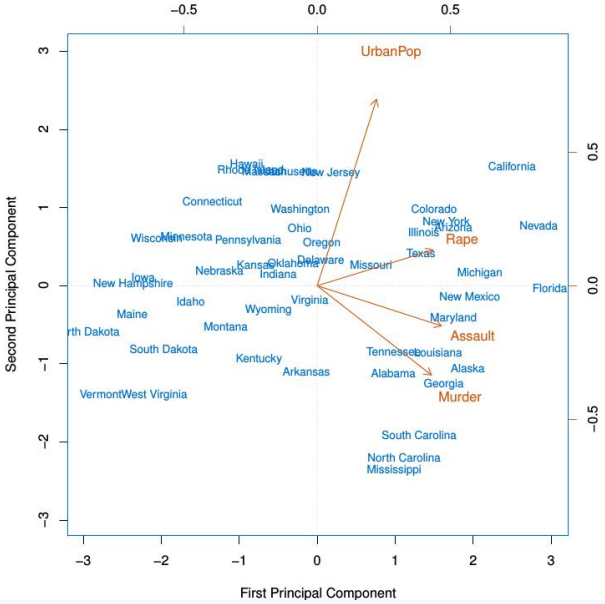
- the loading vectors $\phi_1, \phi_2, \phi_3$ define the ordered orthogonal directions in feature space along which the data vary the most
- if we project the $n$ data points $X_1, \ldots, X_n$ onto this directions $\phi_j$, we obtain the scores $z_{1j}, \ldots, z_{nj}$



## Illustration (taken from Hastie and Tibshirani)

- USAarrests data: For each of the fifty states in the United States, the data set contains the number of arrests per 100,000 residents for each of three crimes: Assault, Murder, and Rape. We also record UrbanPop (the percent of the population in each state living in urban areas).
- The principal component score vectors have length $n = 50$, and the principal component loading vectors have length $p = 4$
- PCA was performed after standardizing each variable to have mean zero and standard deviation one.

# PCA plot for USAarrests data




The first two principal components for the USArrests data.

- The blue state names represent the scores for the first two principal components.
- The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for Rape on the first component is 0.54, and its loading on the second principal component 0.17
- This figure is known as a biplot, because it displays both the principal component scores and the principal component loadings.

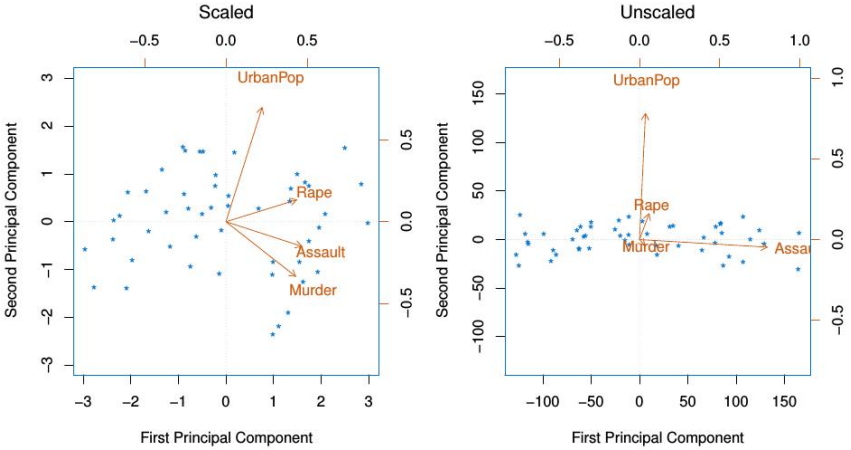
# USAarrests data: PCA loadings

| | PC1 | PC2 |
|---|---|---|
| Murder | 0.5359 | -0.4182 |
| Assault | 0.5832 | -0.1880 |
| UrbanPop | 0.2782 | 0.8728 |
| Rape | 0.5434 | 0.1673 |

- first loading vector $\approx$ measure of overall rate of violence
- second loading vector $\approx$ measure of the level of urbanization

## Scaling of the variables matters

- If the variables are in different units, scaling each to have standard deviation equal to one is recommended.
- If they are in the same units, you might or might not scale the variables.



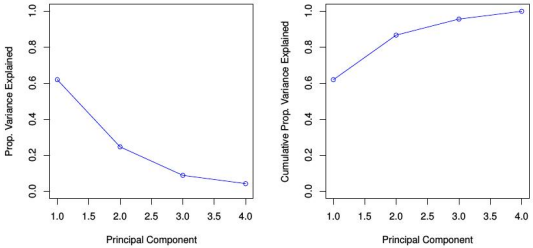
## Proportion Variance Explained

To understand the strength of each component, we are interested in knowing the proportion of variance explained (PVE) by each one.

- The PVE of the $d$th principal component is given by the positive quantity between 0 and 1

$$\frac{\lambda_d}{\sum_{i=1}^{p} \lambda_i}$$

- The PVEs sum to one. We sometimes display the cumulative PVEs: $\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$




☞ How many components should we used? The above scree plots can be used as a guide (look for an elbow)

## PCA in practice

1. Empirical estimation of the mean value:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

2. Empirical estimation of the covariance matrix:

$$\boldsymbol{S} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})(X_i - \overline{X})^\top$$

3. Compute $d$ first eigenvalues/eigenvectors... How to choose $d$? Explained variance:

$$\frac{\sum_{i=1}^{d}\lambda_i}{\sum_{i=1}^{p}\lambda_i}$$

*Note*: Standardization/scaling matters!

Notebooks

# Outline

## PCA for high dimensional data ($n < p$)

$$\mathbf{X} = \begin{bmatrix} (X_1 - \mathbf{m})^T \\ (X_2 - \mathbf{m})^T \\ \vdots \\ (X_n - \mathbf{m})^T \end{bmatrix} \in \mathbb{R}^{n \times p} \quad \Rightarrow \quad \mathbf{S} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{p \times p}$$

Eigendecomposition equation

$$\begin{aligned} \mathbf{S}\mathbf{v}_i &= \lambda_i \mathbf{v}_i, \\ \tfrac{1}{n-1}\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{v}_i &= \lambda_i \mathbf{X}\mathbf{v}_i, \quad \text{(multiplying by } \mathbf{X}\text{)} \\ \Gamma \mathbf{u}_i &= \lambda_i \mathbf{u}_i, \end{aligned}$$

where $\mathbf{u}_i = \mathbf{X}\mathbf{v}_i$ and $\Gamma = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{n \times n}$

- ☞ eigenproblem in $O(pn^2)$ instead of $O(np^2)$
- 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, (B. Schoelkopf, 1998))

## Principle

- Extends conventional principal component analysis (PCA) to a high dimensional feature space using the "kernel trick"
- Can extract up to $n$ (number of samples) nonlinear principal components from a NONLINEAR feature space
  - without expensive computations.
  - dimension reduction will be non-linear in the original space (see general section about Kernel methods)

Let $k$ be a positive semi-definitive kernel function

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

Assume that

$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{x}_i) = \mathbf{0}$$

(*This assumption will be rediscussed later*)

## Kernel PCA equations

$$\Gamma = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \leftarrow C = \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_i)$$

New eigenproblem :

$$C\mathbf{v}_k = \lambda_k \mathbf{v}_k \Leftrightarrow \frac{1}{n}\sum_{i} \Phi(\mathbf{x}_i)[\Phi(\mathbf{x}_i)^T \mathbf{v}_k] = \lambda_k \mathbf{v}_k$$

thus, for $\lambda_k \neq 0$, $\mathbf{v}_k = \sum_i a_{ki}\Phi(\mathbf{x}_i)$ where $a_{ki} = \frac{1}{\lambda_k}\Phi(\mathbf{x}_i)^T \mathbf{v}_k$:

$$\mathbf{v}_k \in \mathrm{span}[\Phi(\mathbf{x}_i), i = 1 \dots n]$$

The eigenequation becomes

$$\left(\frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T\right)\left(\sum_i a_{ki}\Phi(\mathbf{x}_i)\right) = \lambda_k \sum_i a_{ki}\Phi(\mathbf{x}_i)$$

## Kernel PCA equations (Cont'd)

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

$$\frac{1}{n}\sum_i k(\mathbf{x}_i,\mathbf{x}_k)^T \sum_i a_{ki} k(\mathbf{x}_i,\mathbf{x}_k) = \lambda_k \sum_i a_{ki} k(\mathbf{x}_i,\mathbf{x}_k)$$

or in matrix form, letting $[K]_{ik} = k(\mathbf{x_i},\mathbf{x_k}), \forall (i,k) \in [1,..,n]^2$

$$\begin{aligned} & K^2\mathbf{a}_k = \lambda_k n K \mathbf{a}_k \\ \Rightarrow \quad & K\mathbf{a}_k = \lambda_k n \mathbf{a}_k \end{aligned}$$

In practice:

- normalization
$$1 = \mathbf{v}_k^T\mathbf{v}_k = \mathbf{a}_k^T K \mathbf{a}_k = \lambda_k n \mathbf{a}_k^T \mathbf{a}_k$$
- Projection of a new point $\mathbf{x}$ on a principal component $\mathbf{v}_k$:
$$\mathbf{x}^T\mathbf{v}_k = \sum_{i=1}^n a_{ki}\phi(\mathbf{x})^T\phi(\mathbf{x}_i) = \sum_{i=1}^n a_{ki} k(\mathbf{x},\mathbf{x}_i)$$

## Normalizing the NL feature space

- In general, $\Phi(\mathbf{x})$ is not zero mean
- Centered new nonlinear features :

$$\tilde{\Phi}(\mathbf{x}) = \phi(\mathbf{x}) - \mathbf{m}$$

- Introduce the new Kernel

$$\begin{aligned}
\tilde{k}(\mathbf{x_k}, \mathbf{x_l}) &= \tilde{\Phi}(\mathbf{x_k})^T \tilde{\Phi}(\mathbf{x_l}) \\
&= \Phi\left(\mathbf{x_k} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{x_i}\right)^T \Phi\left(\mathbf{x_l} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{x_i}\right) \\
&= k(\mathbf{x_k}, \mathbf{x_l}) - \frac{1}{n}\sum_{i=1}^{n} k(\mathbf{x_k}, \mathbf{x_i}) - \frac{1}{n}\sum_{i=1}^{n} k(\mathbf{x_l}, \mathbf{x_i}) \\
&\quad + \frac{1}{n^2}\sum_{i,j=1}^{n} k(\mathbf{x_i}, \mathbf{x_j}) \\
\Rightarrow \quad \tilde{K} &= K - 2\mathbb{I}_{1/n}K + \mathbb{I}_{1/n}K\mathbb{I}_{1/n}
\end{aligned}$$

where $\mathbb{I}_{1/n} = (\frac{1}{n}, \ldots, \frac{1}{n}) \in \mathbb{R}^n$.

## Summary of kernel PCA

- Choose a Kernel
- Construct the normalized Kernel Matrix

$$\tilde{K} = K - 2\mathbb{I}_{1/n}K + \mathbb{I}_{1/n}K\mathbb{I}_{1/n}$$

- Solve the eigenvalue problem

$$\tilde{K}\mathbf{a}_i = \lambda_i \mathbf{a}_i$$

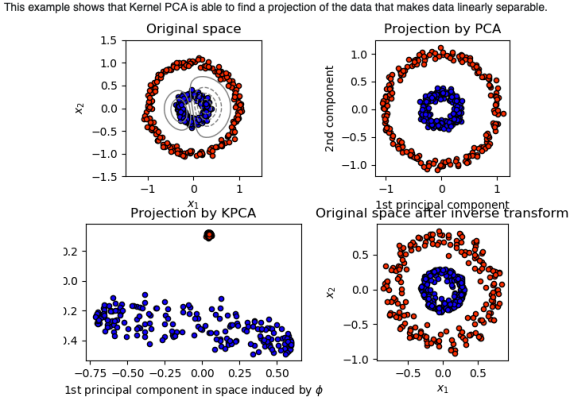
- Any point $\mathbf{x}$ (new or from the initial set of $n$ samples) is represented by its coordinates $z_j$ on the $j-th$ principal axis

$$z_j = \sum_{n=1}^{N} a_{jn} k(\mathbf{x}, \mathbf{x}_n)$$

### Remarks

KPCA gives a good representation of the data when it lies on a NL manifold
$\Rightarrow$ ... but as $K$ is $n \times n$, KPCA may have computational difficulties when $n$ is (too) large.

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




Notebook

# Outline

## Conclusion for PCA

- PCA is a way to explain and understand the variation of an unlabeled dataset $\rightarrow$ useful visualization tool and data preprocessor for supervised learning
- More difficult problem than supervised learning: no ground truth or gold standard (like a response variable) to compare the results, and no single objective (like test set accuracy)
- Kernel PCA: kernel trick can be applied to extend PCA to nonlinear transformation ,
- Active field of research, with many recently developed tools that extends PCA (KPCA, ISOMAP,. . . )