ECE 595: Machine Learning I
Lecture 07 Feature Analysis via PCA

Spring 2020

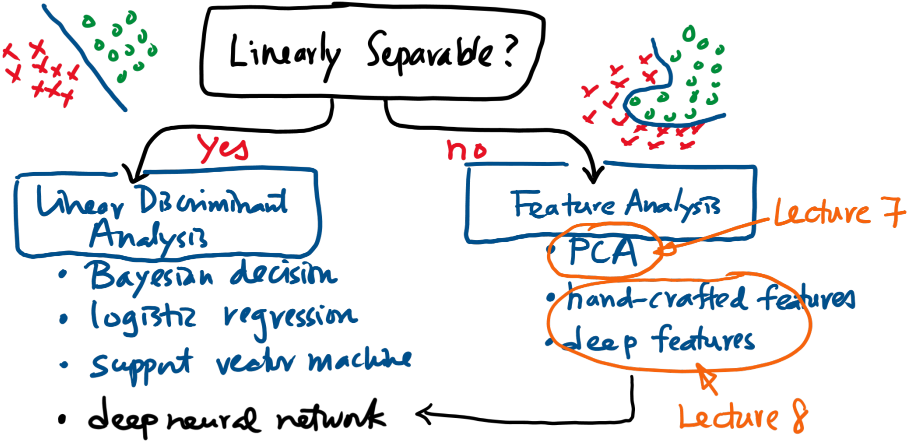
Stanley Chan

School of Electrical and Computer Engineering
Purdue University

PURDUE
UNIVERSITY

Overview

Supervised Learning for Classification



Outline

Feature Analysis

- Lecture 7 Principal Component Analysis (PCA)
- Lecture 8 Hand-Crafted and Deep Features

This Lecture

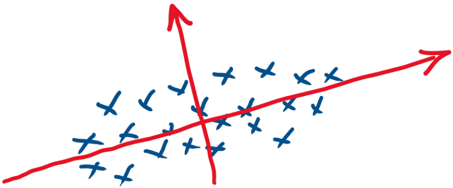
- PCA
  - Low-dimensional Representation
  - Geometric Interpretation
  - Eigen-Face Problem
- Kernel-PCA
  - Adding kernels to PCA
  - Algorithm
  - Examples

Low-Dimensional Representation

- Consider a set of data point $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$
- These data points are living in a **high dimensional space** $\boldsymbol{x}^{(n)} \in \mathbb{R}^d$
- Find a **low dimensional representation** in $\mathbb{R}^p$ where $p < d$
- Equivalent to finding the **principal components** $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ such that

$$\boldsymbol{x}^{(n)} \approx \sum_{i=1}^{p} \alpha_i^{(n)} \boldsymbol{v}_i$$

- Then every $\boldsymbol{x}^{(n)} \in \mathbb{R}^d$ can be represented using $\boldsymbol{\alpha}^{(n)} \in \mathbb{R}^p$.



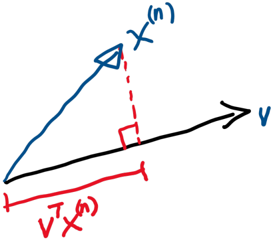
One Sample Analysis

- Consider a simpler problem: One data point $\boldsymbol{x}$ and one direction $\boldsymbol{v}$.
- We want to find a direction $\widehat{\boldsymbol{v}}$ and a scalar $\widehat{\alpha}$ such that

$$(\widehat{\boldsymbol{v}}, \widehat{\alpha}) = \underset{\|\boldsymbol{v}\|_2=1,\alpha}{\text{argmin}} \left\| \begin{bmatrix} | \\ \boldsymbol{x} \\ | \end{bmatrix} - \alpha \begin{bmatrix} | \\ \boldsymbol{v} \\ | \end{bmatrix} \right\|^2$$

- First assume $\boldsymbol{v}$ is available. Then take derivative w.r.t. α:

$$2\boldsymbol{v}^T(\boldsymbol{x} - \alpha\boldsymbol{v}) = 0 \qquad \Rightarrow \qquad \alpha = \boldsymbol{v}^T\boldsymbol{x}.$$



One Sample Analysis

- Substitute $\alpha = \boldsymbol{x}^T\boldsymbol{v}$ into the optimization
- Then the optimization becomes

$$
\begin{aligned}
\underset{\|\boldsymbol{v}\|_2=1}{\text{argmin}} \; \|\boldsymbol{x} - \alpha\boldsymbol{v}\|^2 &= \underset{\|\boldsymbol{v}\|_2=1}{\text{argmin}} \; \left\{ \boldsymbol{x}^T\boldsymbol{x} - 2\alpha\boldsymbol{x}^T\boldsymbol{v} + \alpha^2 \cancel{\boldsymbol{v}^T\boldsymbol{v}} \right\} \\
&= \underset{\|\boldsymbol{v}\|_2=1}{\text{argmin}} \; \left\{ -2\alpha\boldsymbol{x}^T\boldsymbol{v} + \alpha^2 \right\} \\
&= \underset{\|\boldsymbol{v}\|_2=1}{\text{argmin}} \; \left\{ -2(\boldsymbol{x}^T\boldsymbol{v})\boldsymbol{x}^T\boldsymbol{v} + (\boldsymbol{x}^T\boldsymbol{v})^2 \right\} \\
&= \underset{\|\boldsymbol{v}\|_2=1}{\text{argmax}} \; \left\{ \boldsymbol{v}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{v} \right\}
\end{aligned}
$$

- Take expectation on both sides:

$$
\underset{\|\boldsymbol{v}\|_2=1}{\text{argmin}} \; \mathbb{E}_{\boldsymbol{x}}\|\boldsymbol{x} - \alpha\boldsymbol{v}\|^2 = \underset{\|\boldsymbol{v}\|_2=1}{\text{argmax}} \; \boldsymbol{v}^T\mathbb{E}_{\boldsymbol{x}}\left\{ \boldsymbol{x}\boldsymbol{x}^T \right\}\boldsymbol{v}
$$

Eigenvalue Problem

- Let $\boldsymbol{\Sigma} \stackrel{\text{def}}{=} \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T]$.
- Then the optimization problem is

$$\underset{\|\boldsymbol{v}\|_2=1}{\operatorname{argmax}}\, \boldsymbol{v}^T \boldsymbol{\Sigma} \boldsymbol{v}.$$

- The solution to this problem is the eigenvalue and eigenvectors of $\boldsymbol{\Sigma}$.

Theorem

Let $\boldsymbol{\Sigma}$ be a $d \times d$ matrix with eigen-decomposition $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T$. Then, the optimization

$$\widehat{\boldsymbol{v}} = \underset{\|\boldsymbol{v}\|_2=1}{\operatorname{argmax}}\, \boldsymbol{v}^T \boldsymbol{\Sigma} \boldsymbol{v}.$$

has a solution $\widehat{\boldsymbol{v}} = \boldsymbol{u}_i$ for any $i = 1, \ldots, d$.

Proof: See Appendix.

Finite Samples

- When there are N training samples, the optimization is

$$\underset{\|\boldsymbol{v}\|_2=1}{\operatorname{argmin}} \; \underbrace{\frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{x}^{(n)} - \alpha^{(n)}\boldsymbol{v}\|^2}_{=\mathbb{E}[\|\boldsymbol{x}-\alpha\boldsymbol{v}\|^2], \; N\to\infty} = \underset{\|\boldsymbol{v}\|_2=1}{\operatorname{argmax}} \; \boldsymbol{v}^T \underbrace{\left\{\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}(\boldsymbol{x}^{(n)})^T\right\}}_{=\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T], \; N\to\infty}\boldsymbol{v}$$

- In practice, given $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$, we approximate $\boldsymbol{\Sigma}$ by its empirical estimate

$$\boldsymbol{\Sigma} \approx \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}(\boldsymbol{x}^{(n)})^T$$

- You can also remove the mean vectors: $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$:

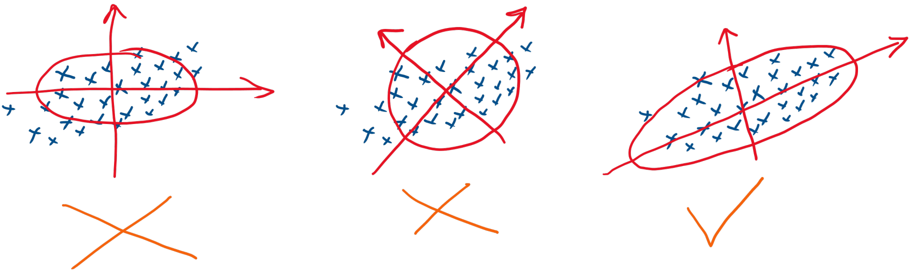
$$\boldsymbol{\Sigma} \approx \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}^{(n)} - \boldsymbol{\mu})(\boldsymbol{x}^{(n)} - \boldsymbol{\mu})^T$$

Statistical Interpretation

- The optimization

$$\underset{\|\boldsymbol{v}\|_2=1}{\operatorname{argmax}}\ \boldsymbol{v}^T \boldsymbol{\Sigma} \boldsymbol{v}.$$

  asks us to find a principal direction that maximizes the **variance**.
- Belief: Large variance = "signal", small variance = "noise"



The Eigenface Problem

Figure: The extended Yale Face Database B.

- Dataset: $\{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$.
- Each $\boldsymbol{x}^{(n)} \in \mathbb{R}^d$ is a vector representation of a $\sqrt{d} \times \sqrt{d}$ image.
- Task 1: Find a low-dimensional representation (This lecture)
- Task 2: Classify faces for a new image (Later)

Low Dimensional Representation

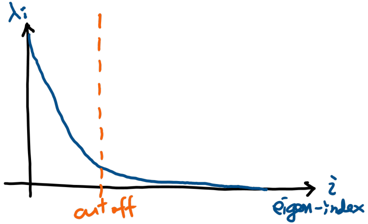
- Estimate the mean vector $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$.
- Estimate the covariance matrix

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)} - \boldsymbol{\mu})(\boldsymbol{x}^{(n)} - \boldsymbol{\mu})^T. \tag{1}$$

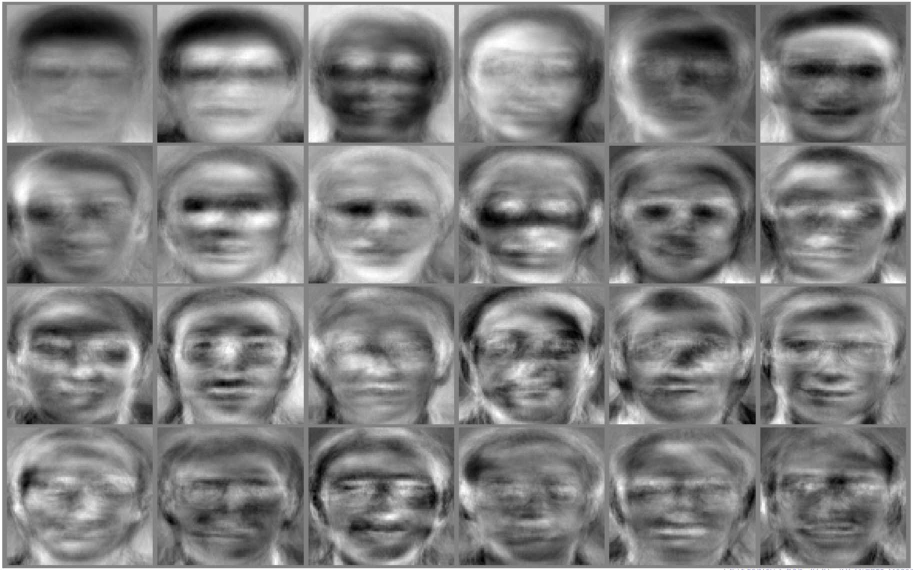
- Eigen-decomposition: $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T$.
- When a new image $\boldsymbol{y}$ comes, estimate the coefficients:

$$\alpha_i = \boldsymbol{u}_i^T \boldsymbol{y}$$

- How many coefficients to use?



The Basis Vectors $\boldsymbol{u}_i$



Representing Faces

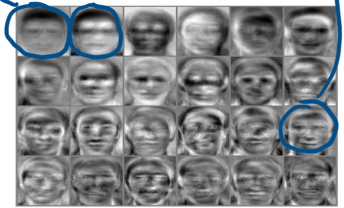
$$\begin{bmatrix} | \\ y \\ | \end{bmatrix} = \alpha_1 \begin{bmatrix} | \\ u_1 \\ | \end{bmatrix} + \alpha_2 \begin{bmatrix} | \\ u_2 \\ | \end{bmatrix} + \ldots \alpha_p \begin{bmatrix} | \\ u_p \\ | \end{bmatrix}$$

data

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_p \end{bmatrix}$$

feature

Basis vectors

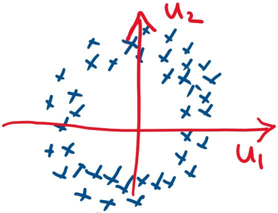


What does PCA do?

- PCA is a tool for **dimension reduction**.
- It compresses a raw data vector $\boldsymbol{y} \in \mathbb{R}^d$ into a smaller feature vector $\boldsymbol{\alpha} \in \mathbb{R}^p$.
- You can now do classification in $\mathbb{R}^p$ instead of $\mathbb{R}^d$.

When will PCA fail?

- When data intrinsically does not have orthogonal projections
- For example, the distributions below



Outline

Feature Analysis

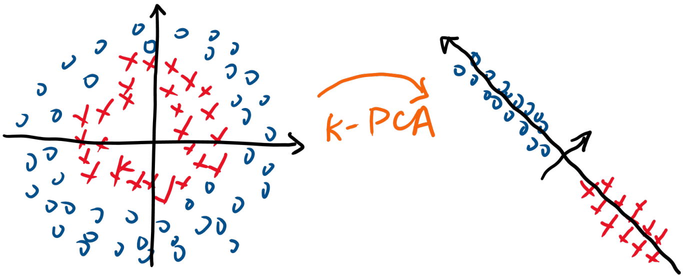
- Lecture 7 Principal Component Analysis (PCA)
- Lecture 8 Hand-Crafted and Deep Features

This Lecture

- PCA
  - Low-dimensional Representation
  - Geometric Interpretation
  - Eigen-Face Problem
- Kernel-PCA
  - Adding kernels to PCA
  - Algorithm
  - Examples

Motivation of Kernel PCA

- Data is originally difficult for PCA
- Find a nonlinear transform
- Idea: Leverage the kernel trick: $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \langle \phi(\boldsymbol{x}^{(i)}), \phi(\boldsymbol{x}^{(j)}) \rangle$
- Example: Left is hard for PCA. After K-PCA, right has a clear principal component.



Kernel for Covariance Matrix

- Assume $\phi(\boldsymbol{x}^{(n)})$ has zero mean. Then consdier the covariance matrix

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}(\boldsymbol{x}^{(n)})^T.$$

- Replacing the outer products by feature transforms

$$\boldsymbol{x}^{(n)} \quad \rightarrow \phi(\boldsymbol{x}^{(n)}),$$

  for some nonlinear transformation ϕ.
- If this can be done, then the covariance will become

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} \phi(\boldsymbol{x}^{(n)})\phi(\boldsymbol{x}^{(n)})^T.$$

- But this is not enough because a kernel needs an inner product

$$k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)}) = \phi(\boldsymbol{x}^{(n)})^T\phi(\boldsymbol{x}^{(m)}).$$

Kernel Trick

- Recall: PCA solves the eigen-decomposition problem:

$$\boldsymbol{\Sigma u} = \lambda \boldsymbol{u}$$

So we also need to consider $\boldsymbol{u}$.

- How about this candidate? (Recall: In Kernel Method we express the model parameter as a linear combination of the samples):

$$\boldsymbol{u} = \sum_{n=1}^{N} \alpha_n \phi(\boldsymbol{x}^{(n)}).$$

- Substitute this into the equation $\boldsymbol{\Sigma u} = \lambda \boldsymbol{u}$:

$$\underbrace{\left(\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}^{(n)})\phi(\boldsymbol{x}^{(n)})^T\right)}_{\boldsymbol{\Sigma}}\underbrace{\left(\sum_{m=1}^{N}\alpha_m\phi(\boldsymbol{x}^{(m)})\right)}_{\boldsymbol{u}} = \underbrace{\lambda\left(\sum_{n=1}^{N}\alpha_n\phi(\boldsymbol{x}^{(n)})\right)}_{\lambda\boldsymbol{u}}$$

Kernel Trick

- This means

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}^{(n)})\left(\sum_{m=1}^{N}\alpha_m\phi(\boldsymbol{x}^{(n)})^T\phi(\boldsymbol{x}^{(m)})\right)=\lambda\sum_{n=1}^{N}\alpha_n\phi(\boldsymbol{x}^{(n)})$$

- Recognizing $\phi(\boldsymbol{x}^{(n)})^T\phi(\boldsymbol{x}^{(m)})=k(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(m)})$:

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}^{(n)})\left(\sum_{m=1}^{N}\alpha_n k(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(m)})\right)=\lambda\sum_{n=1}^{N}\alpha_n\phi(\boldsymbol{x}^{(n)})$$

- Multiply $\phi(\boldsymbol{x}^{(\ell)})^T$ on both sides.

$$\frac{1}{N}\sum_{n=1}^{N}k(\boldsymbol{x}^{(\ell)},\boldsymbol{x}^{(n)})\left(\sum_{m=1}^{N}\alpha_n k(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(m)})\right)=\lambda\sum_{n=1}^{N}\alpha_n k(\boldsymbol{x}^{(\ell)},\boldsymbol{x}^{(n)})$$

- This is $\frac{1}{N}\boldsymbol{K}(\boldsymbol{K}\boldsymbol{\alpha})=\lambda\boldsymbol{K}\boldsymbol{\alpha}$.

Eigenvectors of K-PCA

- Rearrange the terms we have that $\boldsymbol{K}^2\boldsymbol{\alpha} = N\lambda\boldsymbol{K}\boldsymbol{\alpha}$.
- We can remove one of the $\boldsymbol{K}$'s since it only causes issues with zero-eigenvalues which are not important to us anyway. So we have

$$\boldsymbol{K}\boldsymbol{\alpha} = N\lambda\boldsymbol{\alpha}. \tag{2}$$

- This is just another eigen-decomposition problem. We moved from $\boldsymbol{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}$ to $\boldsymbol{K}\boldsymbol{\alpha} = N\lambda\boldsymbol{\alpha}$. Note that $\boldsymbol{\alpha}$ is the coefficients for $\boldsymbol{u}$:

$$\boldsymbol{u} = \sum_{n=1}^{N} \alpha_n \phi(\boldsymbol{x}^{(n)}) = \boldsymbol{\Phi}\boldsymbol{\alpha},$$

where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}^{(1)}), \ldots, \phi(\boldsymbol{x}^{(N)})]$ is the transformed data matrix. Recall $\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \boldsymbol{K}$ is the kernel matrix where

$$[\boldsymbol{K}]_{ij} = \phi(\boldsymbol{x}^{(i)})^T\phi(\boldsymbol{x}^{(j)}).$$

Representation in Kernel Space

- If you run eigen-decomposition on $\boldsymbol{K}$, you will get p eigen-vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p$ where p is the number you choose.
- When a new sample $\boldsymbol{x}$ comes, the j-th representation coefficient is

$$\beta_j = \phi(\boldsymbol{x})^T \boldsymbol{u} = \phi(\boldsymbol{x})^T \sum_{n=1}^{N} \alpha_{jn} \phi(\boldsymbol{x}^{(n)}) = \sum_{n=1}^{N} \alpha_{jn} k(\boldsymbol{x}, \boldsymbol{x}^{(n)}). \tag{3}$$

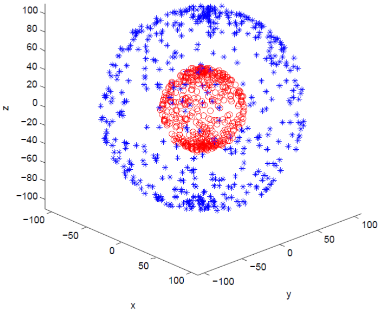
- For the entire representation $\boldsymbol{\beta} \in \mathbb{R}^p$, we have

$$\boldsymbol{\beta} = \begin{bmatrix} ---\boldsymbol{\alpha}_1^T--- \\ \vdots \\ ---\boldsymbol{\alpha}_p^T--- \end{bmatrix} \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}^{(1)}) \\ k(\boldsymbol{x}, \boldsymbol{x}^{(2)}) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}^{(N)}) \end{bmatrix} \tag{4}$$

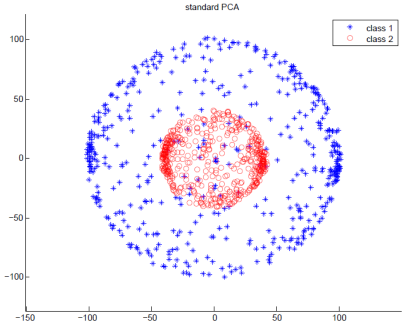
where $\boldsymbol{\alpha}_j = [\alpha_{j1}, \ldots, \alpha_{iN}]^T$.

Example

Here is an example taken from Wang (2012) Kernel Principal Component Analysis and its Applications https://arxiv.org/abs/1207.3538



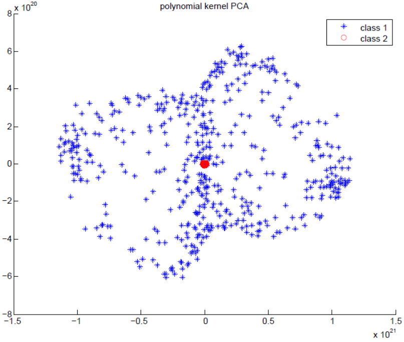
Original Data



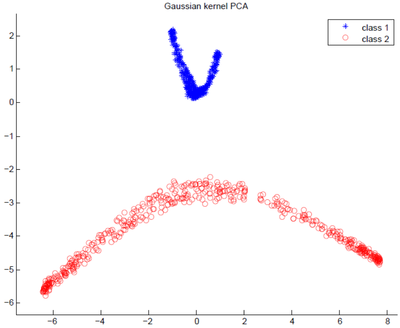
Linear PCA

Example

Here is an example taken from Wang (2012) Kernel Principal Component Analysis and its Applications https://arxiv.org/abs/1207.3538



K-PCA with polynomial



K-PCA with Gaussian

Reading List

PCA Tutorial

- Jonathon Shlens "A Tutorial on Principal Component Analysis", https://arxiv.org/pdf/1404.1100.pdf

PCA: Should We Remove Mean?

- Paul Honeine, "An eigenanalysis of data centering in machine learning", https://arxiv.org/pdf/1407.2904.pdf
- Does mean centering or feature scaling affect a Principal Component Analysis? https://sebastianraschka.com/faq/docs/pca-scaling.html

K-PCA

- Quan Wang (2012), "Kernel Principal Component Analysis and its Applications", https://arxiv.org/abs/1207.3538
- Schölkopf et al. (2005), "Kernel Principal Component Analysis", https://link.springer.com/chapter/10.1007/BFb0020217

Appendix

Proof of Eigenvalue Problem

We want to prove that the solution to the problem

$$\widehat{\boldsymbol{v}} = \underset{\|\boldsymbol{v}\|_2=1}{\text{argmax}}\, \boldsymbol{v}^T \boldsymbol{\Sigma} \boldsymbol{v}.$$

is the eigenvector of the matrix $\boldsymbol{\Sigma}$. To show that, we first write down the Lagrangian:

$$L(\boldsymbol{v}, \lambda) = \boldsymbol{v}^T \boldsymbol{\Sigma} \boldsymbol{v} - \lambda(\|\boldsymbol{v}\|^2 - 1)$$

Take derivative w.r.t. $\boldsymbol{v}$ and setting to zero yields

$$\nabla_{\boldsymbol{v}} L(\boldsymbol{v}, \lambda) = 2\boldsymbol{\Sigma}\boldsymbol{v} - 2\lambda\boldsymbol{v} = \boldsymbol{0}.$$

This is equivalent to $\boldsymbol{\Sigma}\boldsymbol{v} = \lambda\boldsymbol{v}$. So if $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T$, then by letting $\boldsymbol{v} = \boldsymbol{u}_i$ and $\lambda = s_i$ we can satisfy the condition since
$\boldsymbol{\Sigma}\boldsymbol{u}_i = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{u}_i = \boldsymbol{U}\boldsymbol{S}\boldsymbol{e}_i = s_i\boldsymbol{u}_i$.