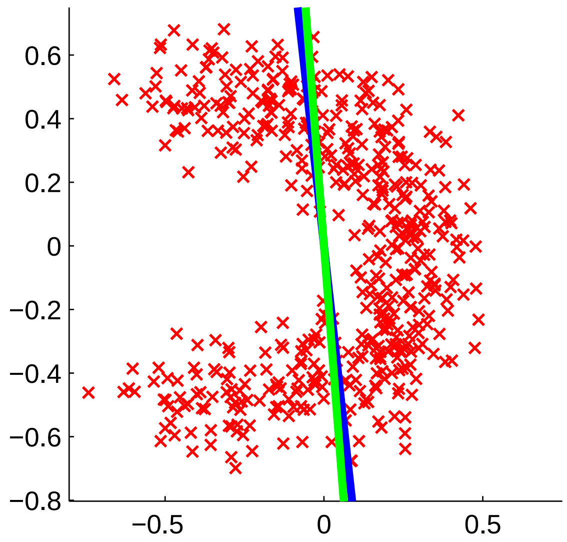
Advanced Data Analysis: Kernel PCA

Masashi Sugiyama (Computer Science)

W8E-505, sugi@cs.titech.ac.jp

http://sugiyama-www.cs.titech.ac.jp/~sugi

Data with Curved Structures

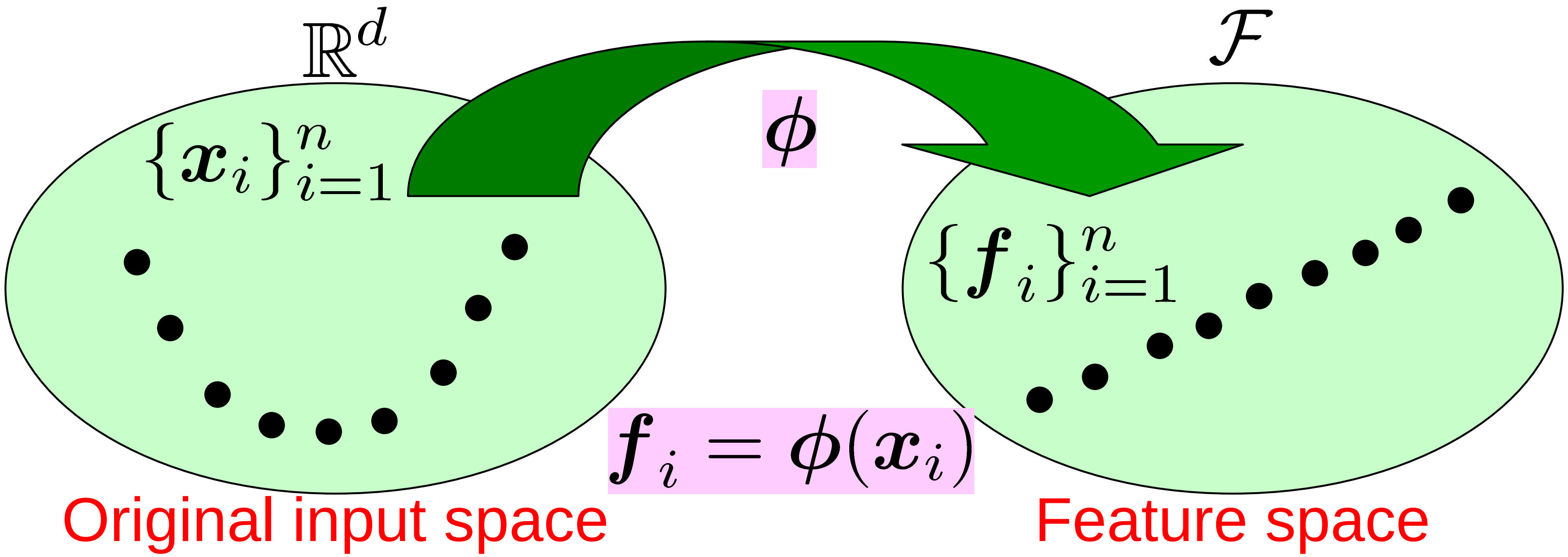


- If the data cloud is bent, any linear methods cannot find the curved structure.

Limitation of linear method!

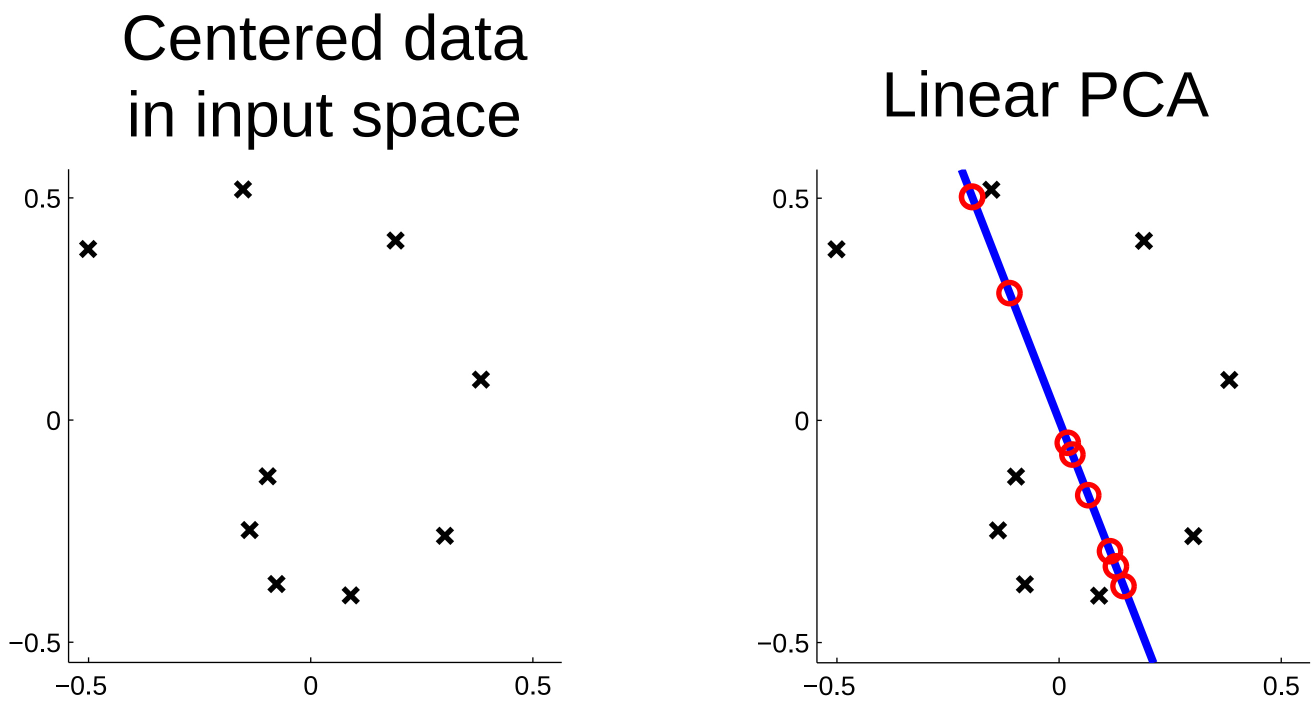
Non-Linearizing Linear Methods

- A simple non-linear extension of linear methods while keeping computational advantages of linear methods:
  - Map the original data to a feature space by a non-linear transformation
  - Run linear algorithm in the feature space



Example

- $d = 2$

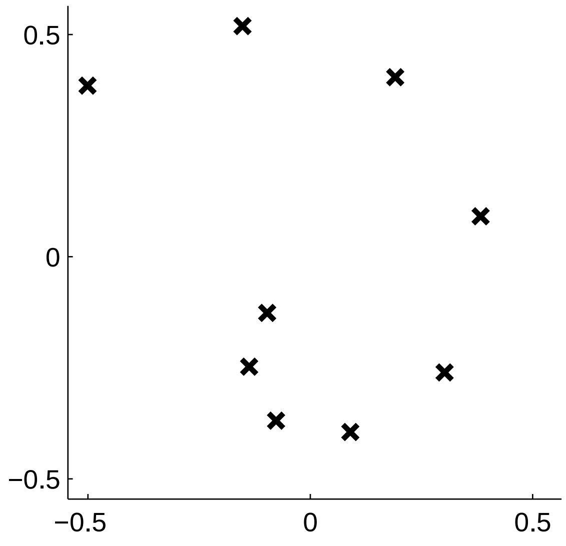


Example (cont.)

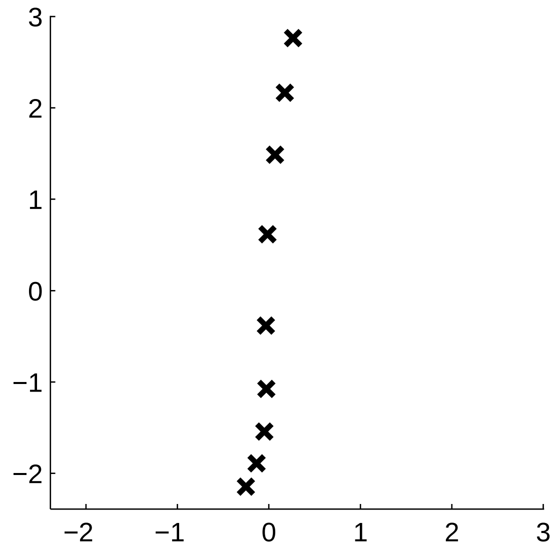
- Polar coordinate:

$$\boldsymbol{x} = \begin{pmatrix} a \\ b \end{pmatrix} \longrightarrow \boldsymbol{f} = \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix}$$

Centered data
in input space

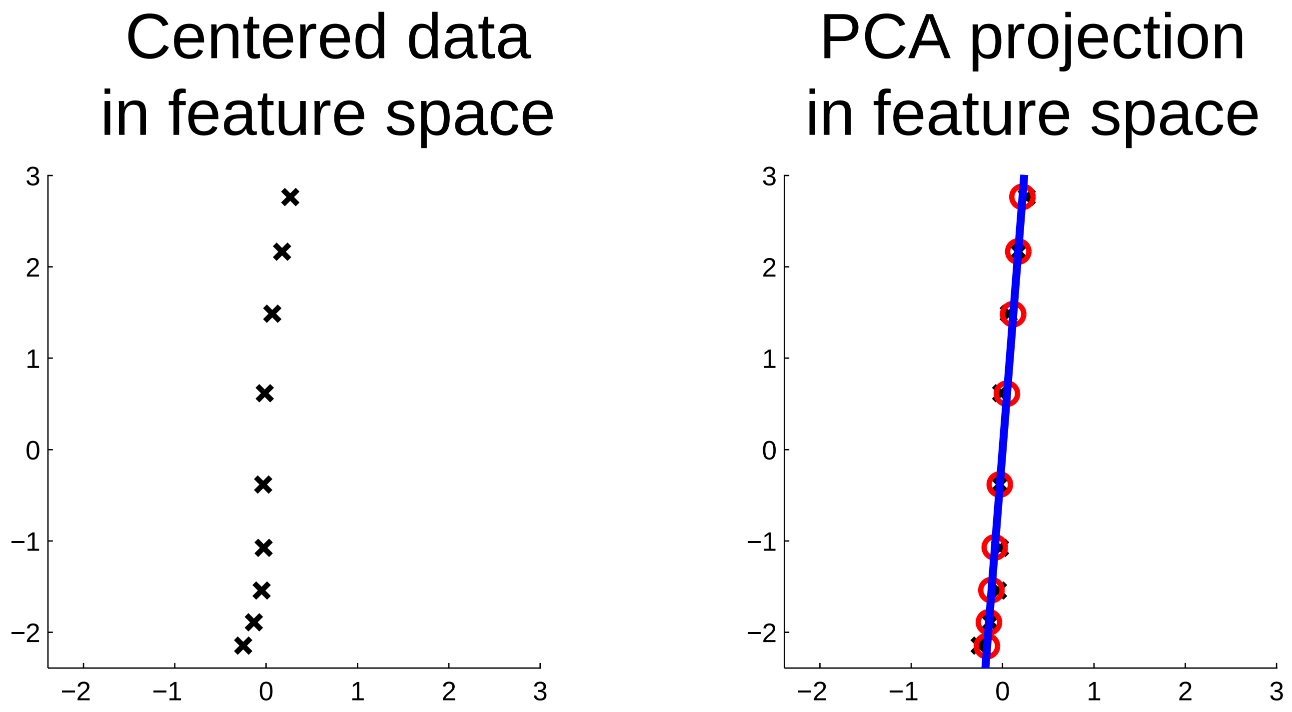


Centered data
in feature space



Example (cont.)

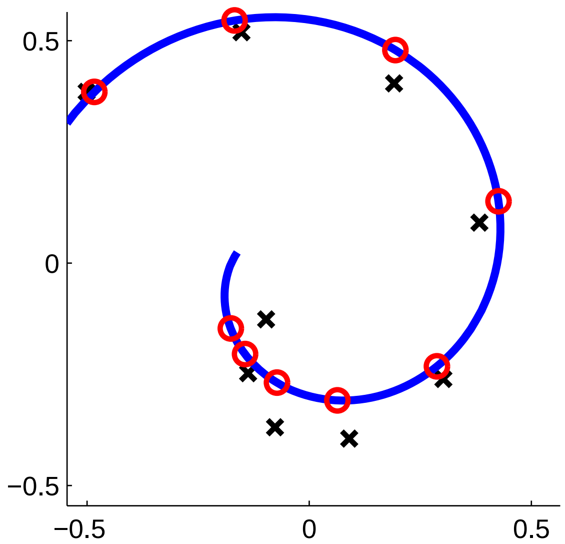
- Run PCA in feature space.



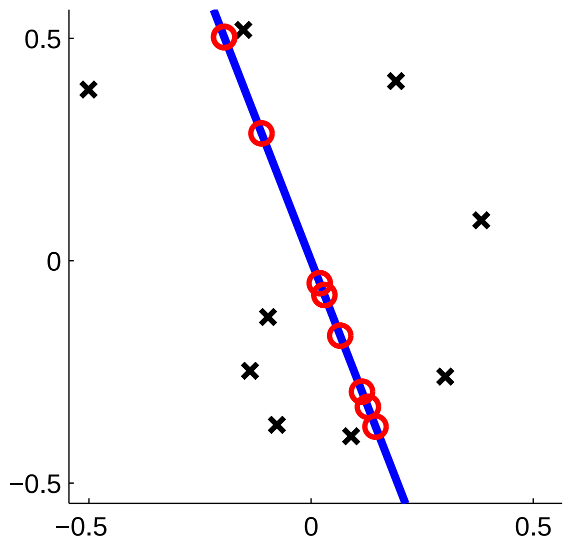
Example (cont.)

- Pull the results back to input space.

Non-linear PCA



Linear PCA



- Non-linear PCA describes the original data much better than linear PCA.

Notation Revisited

- Input samples:

$$\{\boldsymbol{x}_i\}_{i=1}^{n} \qquad \boldsymbol{x}_i \in \mathbb{R}^d$$

- Feature mapping:

$$\boldsymbol{\phi} : \mathbb{R}^d \to \mathcal{F}$$

- Samples in feature space:

$$\boldsymbol{f}_i = \boldsymbol{\phi}(\boldsymbol{x}_i)$$

Centering in Feature Space

- PCA requires centered samples, thus we need to center samples by

$$\overline{\boldsymbol{f}}_i = \boldsymbol{f}_i - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{f}_j$$

- In matrix form,

$$\overline{\boldsymbol{F}} = \boldsymbol{F}\boldsymbol{H}$$

$$\boldsymbol{F} = (\boldsymbol{f}_1|\boldsymbol{f}_2|\cdots|\boldsymbol{f}_n)$$

$$\overline{\boldsymbol{F}} = (\overline{\boldsymbol{f}}_1|\overline{\boldsymbol{f}}_2|\cdots|\overline{\boldsymbol{f}}_n)$$

$$\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_{n\times n}$$

$\boldsymbol{I}_n$: n-dimensional identity matrix

$\mathbf{1}_{n\times n}$: $n \times n$ matrix with all ones

PCA in Feature Space (Primal)

$$\overline{\boldsymbol{C}}\psi = \lambda\psi \qquad \overline{\boldsymbol{C}} = \overline{\boldsymbol{F}}\,\overline{\boldsymbol{F}}^\top$$

- PCA solution:

$$\boldsymbol{B}_{PCA} = (\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2 | \cdots | \boldsymbol{\psi}_m)^\top$$

  - $\{\lambda_i, \boldsymbol{\psi}_i\}_{i=1}^m$:Sorted eigenvalues and normalized eigenvectors of $\overline{\boldsymbol{C}}\psi = \lambda\psi$

$$\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle = \delta_{i,j} \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\mu$$

- PCA embedding of a sample $\boldsymbol{f}$:

$$\overline{\boldsymbol{g}} = \boldsymbol{B}_{PCA}(\boldsymbol{f} - \frac{1}{n}\boldsymbol{F}\boldsymbol{1}_n)$$

$\mu = \dim(\mathcal{F})$

$\boldsymbol{1}_n$: n-dimensional vector with all ones

PCA in High-Dimensional Feature Space

$$\mu = \dim(\mathcal{F})$$

- If μ is high,
  - Description ability of non-linear PCA will increase.
  - However, computational cost increases since the dimension of $\overline{\boldsymbol{C}}$ is μ.
- It would be possible to reduce computational cost since

$$\operatorname{rank}\left(\overline{\boldsymbol{C}}\right) = \min(\mu, n) \leq \mu$$

$$\overline{\boldsymbol{C}} = \overline{\boldsymbol{F}}\,\overline{\boldsymbol{F}}^\top \qquad \overline{\boldsymbol{F}} = (\overline{\boldsymbol{f}}_1 | \overline{\boldsymbol{f}}_2 | \cdots | \overline{\boldsymbol{f}}_n)$$

Dual Formulation

(A) $\overline{\boldsymbol{C}}\psi = \lambda\psi$ $\qquad$ $\overline{\boldsymbol{C}} = \overline{\boldsymbol{F}}\,\overline{\boldsymbol{F}}^\top$

(B) $\overline{\boldsymbol{K}}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$ $\qquad$ $\overline{\boldsymbol{K}} = \overline{\boldsymbol{F}}^\top\overline{\boldsymbol{F}}$

- Solution of (A) can be obtained from (B).
  - Proof: If $\boldsymbol{\alpha}$ is a solution of (B), it holds that

$$\overline{\boldsymbol{C}}\,\overline{\boldsymbol{F}}\boldsymbol{\alpha} = \overline{\boldsymbol{F}\boldsymbol{F}}^\top\,\overline{\boldsymbol{F}}\boldsymbol{\alpha} = \overline{\boldsymbol{F}\boldsymbol{K}}\boldsymbol{\alpha} = \lambda\overline{\boldsymbol{F}}\boldsymbol{\alpha}$$

  This implies that $\psi = \overline{\boldsymbol{F}}\boldsymbol{\alpha}$ is a solution of (A).
- Note: solution of (B) can also be obtained from (A).
- Given $\overline{\boldsymbol{K}}$, solving (B) is faster than (A) when $\mu > n$ since

$$\mathrm{rank}\left(\overline{\boldsymbol{C}}\right) = n < \mu$$

Primal and Dual Formulations

$$\psi = \overline{\boldsymbol{F}}\boldsymbol{\alpha}$$

Primal

$$\overline{\boldsymbol{C}}\psi = \lambda\psi$$

$$\overline{\boldsymbol{C}} = \overline{\boldsymbol{F}}\;\overline{\boldsymbol{F}}^\top$$

Equivalent

Dual

$$\overline{\boldsymbol{K}}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$$

$$\overline{\boldsymbol{K}} = \overline{\boldsymbol{F}}^\top\;\overline{\boldsymbol{F}}$$

$$\overline{\boldsymbol{C}} = \overline{\boldsymbol{F}}\;\overline{\boldsymbol{F}}^\top$$

$$\overline{\boldsymbol{K}} = \overline{\boldsymbol{F}}^\top\overline{\boldsymbol{F}}$$

$$\overline{\boldsymbol{F}} = (\overline{\boldsymbol{f}}_1|\overline{\boldsymbol{f}}_2|\cdots|\overline{\boldsymbol{f}}_n)$$

Renormalization of Eigenvectors

$$\overline{\boldsymbol{K}}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$$

- Standard eigensolvers output an orthonormal eigenvectors.

$$\langle \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j \rangle = \delta_{i,j}$$

- However, PCA requires the primal eigenvectors $\{\boldsymbol{\psi}_i\}_{i=1}^m$ to be orthonormal.
- Since $\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle = \langle \overline{\boldsymbol{K}}\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j \rangle = \lambda_i \delta_{i,j}$, we need to renormalize $\{\boldsymbol{\psi}_i\}_{i=1}^m$ by

$$\boldsymbol{\psi}_i \longleftarrow \frac{\boldsymbol{\psi}_i}{\|\boldsymbol{\psi}_i\|} = \frac{1}{\sqrt{\lambda_i}} \overline{\boldsymbol{F}}\boldsymbol{\alpha}_i$$

$$\boldsymbol{\psi}_i = \overline{\boldsymbol{F}}\boldsymbol{\alpha}_i$$

$$\overline{\boldsymbol{K}}\boldsymbol{\alpha}_i = \lambda_i\boldsymbol{\alpha}_i$$

PCA in Feature Space (Dual)

- PCA embedding of a sample $\boldsymbol{f}$:

$$\overline{\boldsymbol{g}} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{A}^{\top} \boldsymbol{H}(\boldsymbol{k} - \frac{1}{n}\boldsymbol{K}\boldsymbol{1}_n)$$ (Homework)

  - $\{\lambda_i, \boldsymbol{\alpha}_i\}_{i=1}^{m}$:Sorted eigenvalues and normalized eigenvectors of $\overline{\boldsymbol{K}}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \qquad \langle \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j \rangle = \delta_{i,j}$$

$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$

$\boldsymbol{A} = (\boldsymbol{\alpha}_1 | \boldsymbol{\alpha}_2 | \cdots | \boldsymbol{\alpha}_m)$

$\overline{\boldsymbol{K}} = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}$ $\quad$ $\boldsymbol{K} = \boldsymbol{F}^{\top}\boldsymbol{F}$

$\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\boldsymbol{1}_{n\times n}$ $\quad$ $\boldsymbol{k} = \boldsymbol{F}^{\top}\boldsymbol{f}$

$\boldsymbol{I}_n$: n-dimensional identity matrix

$\boldsymbol{1}_{n\times n}$: $n \times n$ matrix with all ones

$\boldsymbol{1}_n$: n-dimensional vector with all ones

PCA in Feature Space (Dual)

$\mu = \dim(\mathcal{F})$

- In the dual formulation, the computational complexity depends not on μ but only on n, if $\boldsymbol{K}$ and $\boldsymbol{k}$ are given.
- However, the computation of $\boldsymbol{K}$ and $\boldsymbol{k}$ still depends on μ.

$$\boldsymbol{K} = \boldsymbol{F}^\top \boldsymbol{F} \qquad \boldsymbol{k} = \boldsymbol{f}^\top \boldsymbol{F}$$

- Note: $\boldsymbol{K}$ and $\boldsymbol{k}$ depend on μ only through the inner product between samples.

$$\boldsymbol{K}_{i,j} = \langle \boldsymbol{f}_i, \boldsymbol{f}_j \rangle \qquad \boldsymbol{k}_i = \langle \boldsymbol{f}, \boldsymbol{f}_i \rangle$$

Kernel Trick

- For some transformation $\phi(\boldsymbol{x})\ \ (=\boldsymbol{f})$, there exists a bivariate function $K(\boldsymbol{x}, \boldsymbol{x}')$ such that

$$\boldsymbol{K}_{i,j} = \langle \boldsymbol{f}_i, \boldsymbol{f}_j \rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Such implicit mapping $\phi(\boldsymbol{x})$ exists if
  - $\boldsymbol{K}$ is symmetric: $\boldsymbol{K}^\top = \boldsymbol{K}$
  - $\boldsymbol{K}$ is positive semi-definite: $\forall \boldsymbol{y},\ \ \langle \boldsymbol{K}\boldsymbol{y}, \boldsymbol{y} \rangle \geq 0$
- Such $K(\boldsymbol{x}, \boldsymbol{x}')$ is called the reproducing kernel.
- Rather than directly defining $\phi(\boldsymbol{x})$, we implicitly specify $\phi(\boldsymbol{x})$ by a reproducing kernel.

Examples of Kernels

$$\mu = \dim(\mathcal{F})$$

- Polynomial kernel:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^c \qquad c \in \mathbb{N}$$

  - When $d = 2$ and $c = 2$,

$$\boldsymbol{x} = \begin{pmatrix} s \\ t \end{pmatrix}$$

$$\begin{aligned} \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2 &= (ss' + tt')^2 \\ &= sss's' + 2ss'tt' + ttt't' \end{aligned}$$

$$\Rightarrow \quad \boldsymbol{f} = \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} s^2 \\ \sqrt{2}st \\ t^2 \end{pmatrix}$$

$$\mu = 3$$

  - In general,

$$\mu = {}_{c+d-1}C_c$$

Examples of Kernels (cont.)

- Gaussian kernel:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2 / c^2\right)$$

$c > 0$

Note: $\mu = \infty$!

$\mu = \dim(\mathcal{F})$

Kernel PCA: Summary

- Kernel PCA embedding of a sample $\boldsymbol{f}$ is

$$\overline{\boldsymbol{g}} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{A}^{\top} \boldsymbol{H}(\boldsymbol{k} - \frac{1}{n} \boldsymbol{K} \boldsymbol{1}_n)$$

  - $\{\lambda_i, \boldsymbol{\alpha}_i\}_{i=1}^{m}$:Sorted eigenvalues and normalized eigenvectors of $\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \qquad \langle \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j \rangle = \delta_{i,j}$$

$\boldsymbol{\Lambda} = \mathrm{diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_m\right)$

$\boldsymbol{A} = (\boldsymbol{\alpha}_1 | \boldsymbol{\alpha}_2 | \cdots | \boldsymbol{\alpha}_m)$

$\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n} \boldsymbol{1}_{n \times n}$

$\boldsymbol{k} = (K(\boldsymbol{x}, \boldsymbol{x}_1), K(\boldsymbol{x}, \boldsymbol{x}_2), \ldots, K(\boldsymbol{x}, \boldsymbol{x}_n))^{\top}$

$\boldsymbol{I}_n$: n-dimensional identity matrix

$\boldsymbol{1}_{n \times n}$: $n \times n$ matrix with all ones

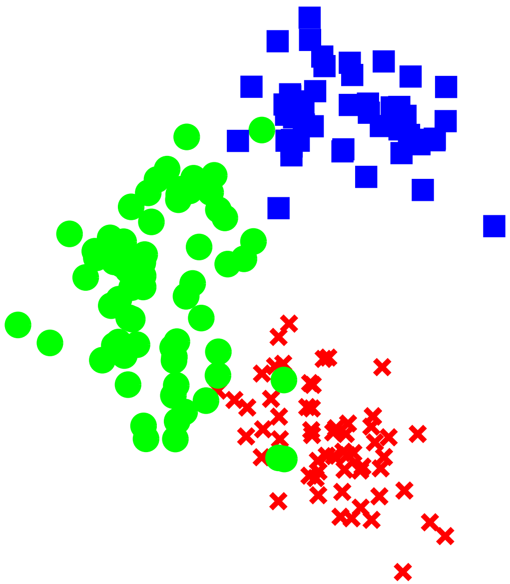
$\boldsymbol{1}_n$: n-dimensional vector with all ones

$\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

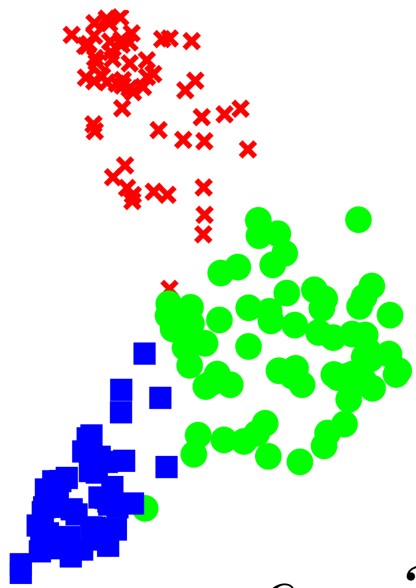
Examples

- Wine data (UCI): 13-dim, 178 samples

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2 / c^2\right)$$



Linear PCA

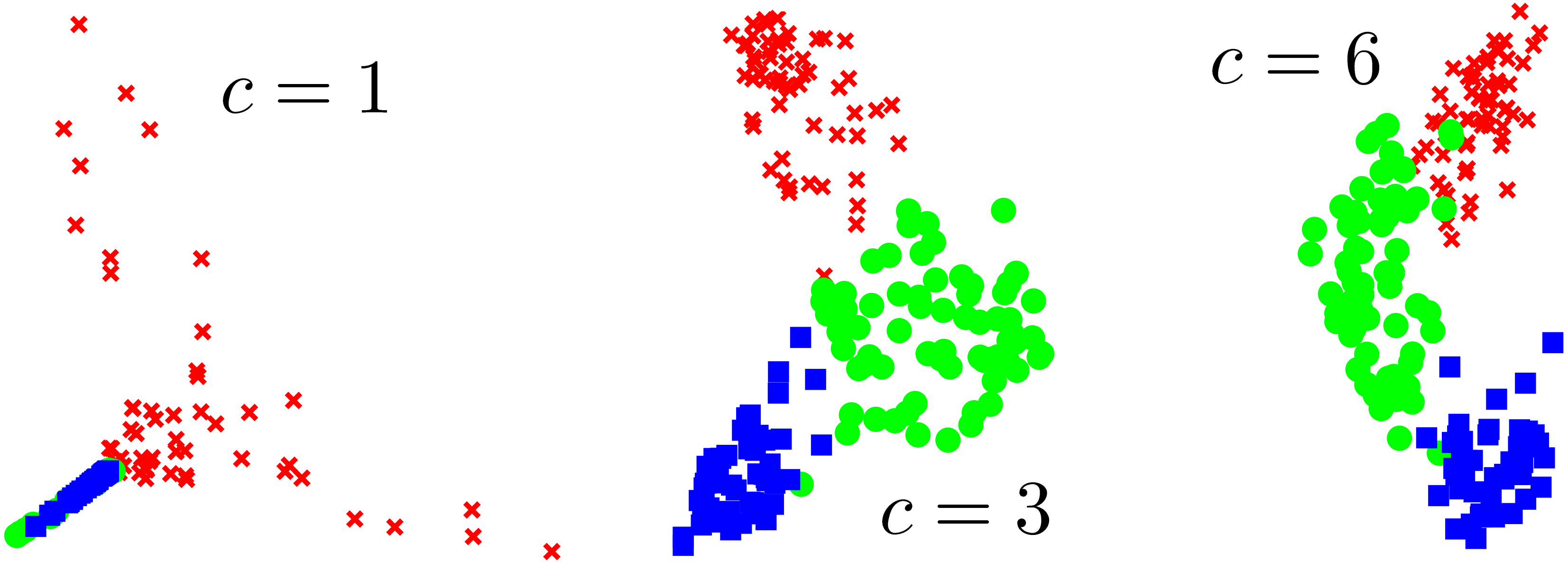


$c = 3$

Gaussian KPCA

Examples (cont.)

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2 / c^2\right)$$



- Choice of kernels (type and parameter) depends on the result.
- Appropriately choosing kernels is not straightforward in practice.

Homework

1. Implement kernel PCA with Gaussian kernels and reproduce the embedding result of the Wine data set.

   http://sugiyama-www.cs.titech.ac.jp/~sugi/data/DataAnalysis

   Test kernel PCA with your own (artificial or real) data and analyze the characteristics of kernel PCA.

2. Prove that kernel PCA embedding of a sample f is given by

$$\overline{\boldsymbol{g}} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{A}^\top \boldsymbol{H} (\boldsymbol{k} - \frac{1}{n} \boldsymbol{K} \boldsymbol{1}_n)$$

Suggestion

- Read the following article for the next class:
  - M. Belkin & P. Niyogi: Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation, 15(6), 1373-1396, 2003.

http://neco.mitpress.org/cgi/reprint/15/6/1373.pdf