Prof. Marios Savvides

Pattern Recognition Theory

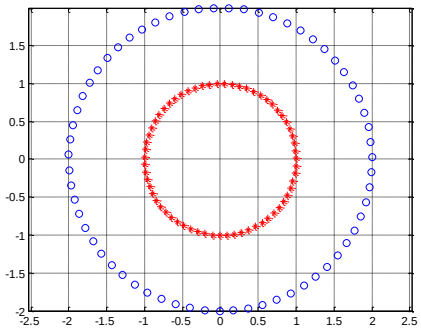
Lecture 14 : Kernel Feature Analysis

All graphics from Pattern Classification, Duda, Hart and Stork, Copyright © John Wiley and Sons, 2001

Electrical & Computer ENGINEERING

Linear vs Nonlinear Methods

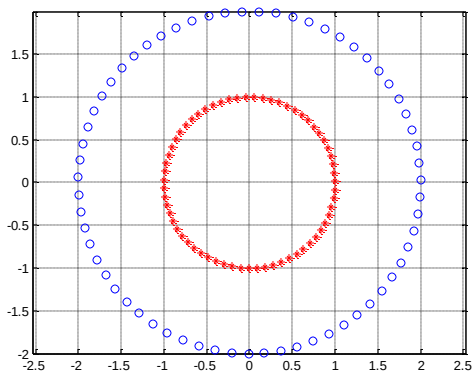
- Linear methods
  - Assume that data are linearly separable.
  - Simple, easy, fundamental, computationally cheap, etc.
  - However, real-world data samples are hardly linearly separable.



- Nonlinear methods
  - Applicable to complex real-world data that are not linearly separable.
  - Provide more classification power.

Linear vs Nonlinear Methods

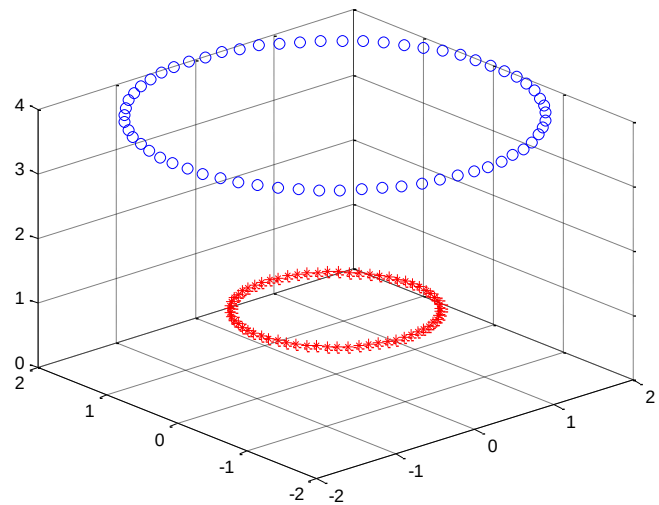
- Data are not linearly separable -> might be separable in high dimension.



original data space
- lower dim.
- not linearly separable

$$\phi: \quad R^2 \rightarrow \quad R^3$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_1^{\ 2} + x_2^{\ 2} \end{bmatrix}$$



feature space
- higher dim.
- linearly separable

- Mappings onto a space higher in dimension than the original space might provide greater classification power.
- Data become linearly separable at the expense of dimensionality.

Explicit Mapping Example

- Consider polynomial kernel

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

  - $d = 1$

$$\Phi(\mathbf{x}) = \mathbf{x} \qquad \Phi(\mathbf{x}) \bullet \Phi(\mathbf{y}) = \left(x_1 y_1 + x_2 y_2\right) = \mathbf{x} \bullet \mathbf{y}$$

  - $d = 2$ $\quad \Phi(\mathbf{x}) = \begin{bmatrix} x_1^2 & x_1 x_2 & x_2 x_1 & x_2^2 \end{bmatrix}^T$

$$\Phi(\mathbf{x}) \bullet \Phi(\mathbf{y}) = \begin{bmatrix} x_1^2 & x_1 x_2 & x_2 x_1 & x_2^2 \end{bmatrix} \begin{bmatrix} y_1^2 \\ y_1 y_2 \\ y_2 y_1 \\ y_2^2 \end{bmatrix} = x_1^2 y_1^2 + 2 x_1 x_2 y_1 y_2 + x_2^2 y_2^2 = \left(\mathbf{x} \bullet \mathbf{y}\right)^2$$

  - Any d

$$\Phi(\mathbf{x}) \bullet \Phi(\mathbf{y}) = \left(\mathbf{x} \bullet \mathbf{y}\right)^d$$

- Define Kernel function

$$K\left(\mathbf{x}, \mathbf{y}\right) = \left(\mathbf{x} \bullet \mathbf{y}\right)^d = \Phi(\mathbf{x}) \bullet \Phi(\mathbf{y})$$

Kernel Methods

- Problem: ϕ is unknown.
  - A proper nonlinear mapping function is not defined explicitly.
  - We do not even know the dimensionality of the feature space.
- Kernel-based extension of a linear methods
  - If a certain linear method is formulated with dot products, it can be performed in the higher dimensional feature space based on $\phi(\mathbf{x}) \bullet \phi(\mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$.
  - Thus, if $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \bullet \phi(\mathbf{y})$ is defined, the linear method can be applied to the feature space without any explicit usage of ϕ.
- Kernel Trick
  - Data are not represented individually anymore, but only through a set of pairwise comparisons.
  - Instead of using a mapping $\phi : R^d \rightarrow R^{d_{high}}$ to represent each object $\mathbf{x} \in R^d$, a comparison function $k(\mathbf{x},\mathbf{y})$ is used.
    - Hence, no need to compute mapping
    - No need to compute dot product

The Kernel Trick

- We can take advantage of high dimensional representations without having to work in the high dimension space.
- It is possible to compute dot products in high dimension spaces **without explicitly mapping** into these spaces. We employ dot products of the form

$$K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x}) \bullet \Phi(\mathbf{y})$$

  - Example: Polynomial kernel of degree d

$$K(\mathbf{x},\mathbf{y}) = \underbrace{\Phi(\mathbf{x}) \bullet \Phi(\mathbf{y})}_{\text{SKIP}} = (\mathbf{x} \bullet \mathbf{y})^d$$

Go directly here

- A Kernel function K() can be represented as
  - $K(\mathbf{x},\mathbf{y})=\Phi(\mathbf{x})^T \Phi(\mathbf{y})$ if and only if for a square integrable function $g(\mathbf{x})$ ($\int g(\mathbf{x})^2 d\mathbf{x}$ is finite

$$\int K(\mathbf{x},\mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

    - This is called **Mercer's condition**.

Common Kernels

- Polynomial of degree *d*

$$K(\mathbf{x},\mathbf{y}) = \left(\mathbf{x} \bullet \mathbf{y}\right)^d$$

- Polynomial of degree up to *d*

$$K(\mathbf{x},\mathbf{y}) = \left(\mathbf{x} \bullet \mathbf{y} + 1\right)^d$$

- Gaussian Kernels (Radial Basis Function)

$$K(\mathbf{x},\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}$$

- Hyperbolic Tangent

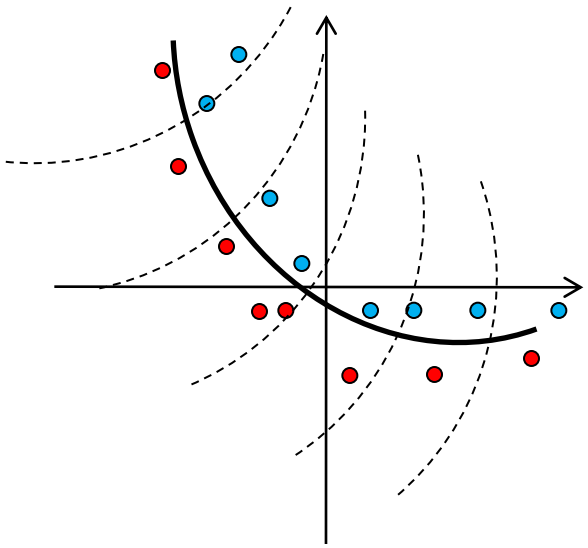
$$K(\mathbf{x},\mathbf{y}) = \tanh\left(\eta \mathbf{x} \bullet \mathbf{y} + v\right)$$

Kernel PCA

- KPCA is designed to perform PCA in the arbitrarily high-dimensional feature space related to the input space through a nonlinear mapping.
- KPCA assumes that the nonlinear mapping makes the mapped data linearly separable.
- Based on the observation that PCA is formulated with dot products, one can apply the kernel trick to PCA and consequently obtain KPCA, the kernel-based extension of PCA.

Kernel PCA

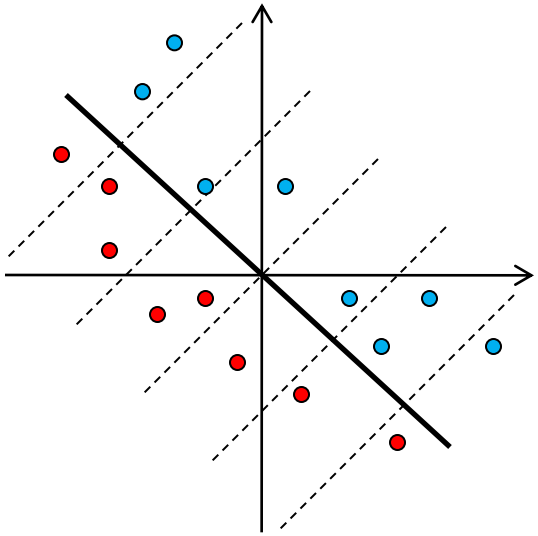
- low dimension: $\mathbf{x} \in R^d$
- nonlinear: $k(\mathbf{x}_i, \mathbf{x}_j)$



$\phi(\cdot)$

$d << d_{high}$

- high dimension: $\phi(\mathbf{x}) \in R^{d_{high}}$
- linear: $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$



Kernel trick: $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle$

The subspace achieved by KPCA is nonlinear in the input space, which allows KPCA to be applied to complex real-world data that are not linearly separable.

PCA Refresher

- PCA efficiently represents the data by finding orthonormal axes which maximally de-correlate the data
  - Assumptions: data samples are independent, and Gaussian distributed.
- PCA finds the principal axes by diagonalizing the covariance matrix.

$$\Sigma = \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathbf{T}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}\mathbf{x_i}^{\mathbf{T}}$$

  - Assume N sample points that have been centered (m = 0)

$$\lambda\mathbf{v} = \mathbf{\Sigma}\mathbf{v}$$

  - Find all eigenvectors, arrange them, project onto them, and use the coefficients

- To be able to use kernels for PCA, we have to rewrite it in terms of dot products

$$\mathbf{\Sigma}\mathbf{v} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}\mathbf{x_i}^{\mathbf{T}}\mathbf{v} = \lambda\mathbf{v}$$

  - Thus

$$\mathbf{v} = \frac{1}{N\lambda}\sum_{i=1}^{N}\mathbf{x_i}\mathbf{x_i}^{\mathbf{T}}\mathbf{v} = \frac{1}{N\lambda}\sum_{i=1}^{N}\left(\mathbf{x_i}\bullet\mathbf{v}\right)\mathbf{x_i}$$

Show that $(xx^T)v=(x^Tv)x$

$$\mathbf{x}\mathbf{x}^{\mathbf{T}}\mathbf{v} = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_d \\ x_2x_1 & x_2x_2 & & \vdots \\ \vdots & & \ddots & \\ x_dx_1 & & & x_dx_d \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix} = \begin{bmatrix} x_1x_1v_1 + x_1x_2v_2 + \cdots + x_1x_dv_d \\ x_2x_1v_1 + x_2x_2v_2 + \cdots + x_2x_dv_d \\ \vdots \\ x_cx_1v_1 + x_cx_2v_2 + \cdots + x_dx_dv_d \end{bmatrix}$$

$$\begin{bmatrix} \left(x_1v_1 + x_2v_2 + \cdots + x_dv_d\right)x_1 \\ \left(x_1v_1 + x_2v_2 + \cdots + x_dv_d\right)x_2 \\ \vdots \\ \left(x_1v_1 + x_2v_2 + \cdots + x_dv_d\right)x_d \end{bmatrix} = \left[x_1v_1 + x_2v_2 + \cdots + x_dv_d\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \left(\mathbf{x}^{\mathbf{T}}\mathbf{v}\right)\mathbf{x}$$

KPCA - 1

- We had

$$\mathbf{v} = \frac{1}{N\lambda}\sum_{i=1}^{N}\mathbf{x_i}\mathbf{x_i}^{\mathbf{T}}\mathbf{v} = \frac{1}{N\lambda}\sum_{i=1}^{N}\left(\mathbf{x_i}\bullet\mathbf{v}\right)\mathbf{x_i}$$

- However ($\mathbf{x_i}.\mathbf{v}$) is just a scalar, which means that all solutions **v** with non zero eigenvalue lie in the span of $x_1 \ldots x_d$

$$\mathbf{v} = \sum_{i=1}^{N}\alpha_i\mathbf{x}_i$$

Eigenvectors can be expressed as linear combinations of the training data

- Map into another space,

$$\Phi : X \rightarrow H \text{ , } \mathrm{x} \rightarrow \Phi(\mathrm{x})$$

- Assuming we can center the data in that space ($\sum_{k=1}^{N}\Phi\left(\mathbf{x}_k\right) = 0$), we can write the covariance matrix as:

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}\Phi\left(\mathbf{x_i}\right)\Phi\left(\mathbf{x_i}\right)^{\mathbf{T}}$$

- We just showed that all solutions v with non zero eigenvalues must lie in the span of Phi(x_1)….Phi(x_N), that is

$$\mathbf{\Sigma v} = \lambda\mathbf{v} = \lambda\sum_{i=1}^{N}\alpha_i\Phi\left(\mathbf{x_i}\right)$$

KPCA - 2

- Proof

$$\Phi : X \rightarrow H \text{ , } \mathrm{x} \rightarrow \Phi(\mathrm{x}) \qquad \mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}\Phi\left(\mathbf{x_i}\right)\Phi\left(\mathbf{x_i}\right)^{\mathbf{T}}$$

$$\mathbf{\Sigma v} = \lambda \mathbf{v} \rightarrow \left(\frac{1}{N}\sum_{i=1}^{N}\Phi\left(\mathbf{x_i}\right)\Phi\left(\mathbf{x_i}\right)^{\mathbf{T}}\right)\mathbf{v} = \lambda \mathbf{v}$$

$$\mathbf{v} = \left(\frac{1}{\lambda N}\sum_{i=1}^{N}\Phi\left(\mathbf{x_i}\right)\Phi\left(\mathbf{x_i}\right)^{\mathbf{T}}\right)\mathbf{v} = \sum_{i=1}^{N}\left(\frac{\Phi\left(\mathbf{x_i}\right)^{T}\mathbf{v}}{\lambda N}\right)\Phi\left(\mathbf{x_i}\right) = \sum_{i=1}^{N}\alpha_i\Phi\left(\mathbf{x_i}\right)$$

- To express the relationship entirely in terms of the inner-product kernel, we premultiply both sides by $\Phi\left(\mathbf{x}_k\right)^T$

$$\Phi\left(\mathbf{x}_k\right)^T \mathbf{\Sigma v} = \lambda \Phi\left(\mathbf{x}_k\right)^T \mathbf{v}$$

- Combine everything back into the previous equation

$$\lambda\left[\Phi\left(\mathbf{x}_k\right)^T \sum_{i=1}^{N}\alpha_i\Phi\left(\mathbf{x_i}\right)\right] = \Phi\left(\mathbf{x}_k\right)^T\left[\frac{1}{N}\sum_{j=1}^{N}\Phi\left(\mathbf{x_j}\right)\Phi\left(\mathbf{x_j}\right)^{\mathbf{T}}\right]\left[\sum_{i=1}^{N}\alpha_i\Phi\left(\mathbf{x_i}\right)\right]$$

KPCA - 3

- Regrouping terms, we obtain

$$\lambda\sum_{i=1}^{N}\alpha_i\left(\Phi\left(\mathbf{x}_k\right)^T\Phi\left(\mathbf{x_i}\right)\right)=\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\Phi\left(\mathbf{x}_k\right)^T\Phi\left(\mathbf{x}_i\right)\left(\Phi\left(\mathbf{x}_i\right)^{\mathbf{T}}\Phi\left(\mathbf{x}_j\right)\right)$$

- Define the *NxN* matrix **K**, called the kernel matrix, whos ijth element in the inner product kernel K($\mathbf{x_i}$,$\mathbf{x_j}$)
- The Nx1 vector **α** whose jth element is the coefficient α_j

$$\mathbf{K}=\begin{bmatrix} K\left(\mathbf{x_1},\mathbf{x_1}\right) & \cdots & K\left(\mathbf{x_1},\mathbf{x_N}\right) \\ \vdots & \ddots & \vdots \\ K\left(\mathbf{x_N},\mathbf{x_1}\right) & \cdots & K\left(\mathbf{x_N},\mathbf{x_N}\right) \end{bmatrix} \qquad \boldsymbol{\alpha}=\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} \qquad K\left(\mathbf{x_i},\mathbf{x_j}\right)=\left\langle\Phi\left(\mathbf{x_i}\right),\Phi\left(\mathbf{x_j}\right)\right\rangle$$

- The first equation becomes:

$$N\lambda\mathbf{K}\boldsymbol{\alpha}=\mathbf{K}^2\boldsymbol{\alpha}$$

  – Which can be solved by the following eigenvalue/eigenvector problem:

$$\boxed{N\lambda\boldsymbol{\alpha}=\mathbf{K}\boldsymbol{\alpha}}$$

KPCA - 4

- Normalization
  - We have to make sure than the eigenvectors V are orthonormal, we scale eigenvectors **α**

$$\left\langle \mathbf{v}^{(k)}, \mathbf{v}^{(k)} \right\rangle = 1 \Rightarrow \left(\sum_{i=1}^{N} \alpha_i^{(k)} \Phi\left(\mathbf{x}_i\right) \right) \left(\sum_{j=1}^{N} \alpha_j^{(k)} \Phi\left(\mathbf{x}_j\right) \right) = 1$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^{(k)} \alpha_j^{(k)} \Phi\left(\mathbf{x}_i\right) \Phi\left(\mathbf{x}_j\right) = 1 \Rightarrow$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^{(k)} \alpha_j^{(k)} \mathbf{K_{ij}} = 1 \Rightarrow \left(\alpha^{(k)} \mathbf{K} (\alpha^{(k)})^T \right) = 1$$

- Projecting a new test sample:
  - We now show how to project a test point onto the eigenvectors in the high-dimensional space H in terms of dot product (so we can still utilize the kernel trick).

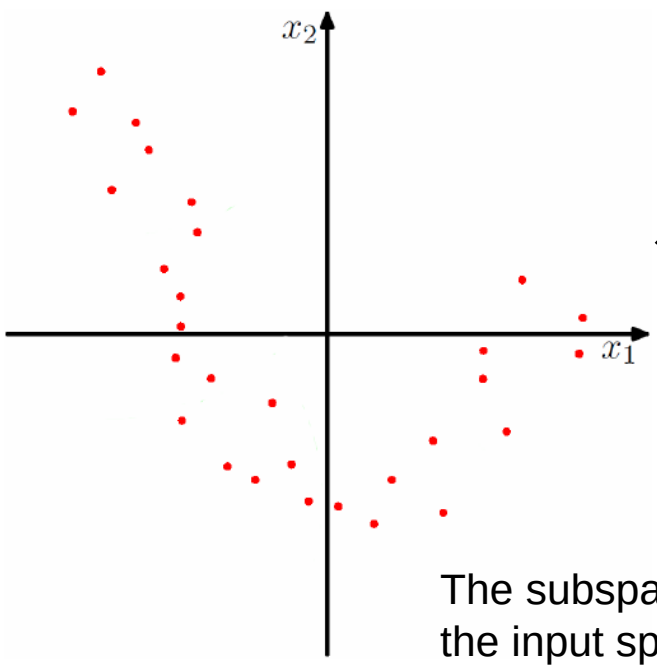
$$\left\langle \mathbf{v}^{(k)}, \Phi\left(\mathbf{x}\right) \right\rangle = \left(\sum_{i=1}^{N} \alpha_i^{(k)} \Phi\left(\mathbf{x_i}\right) \right) \bullet \Phi\left(\mathbf{x}\right) = \sum_{i=1}^{N} \alpha_i^{(k)} \mathbf{K}\left(\mathbf{x_i}, \mathbf{x}\right)$$

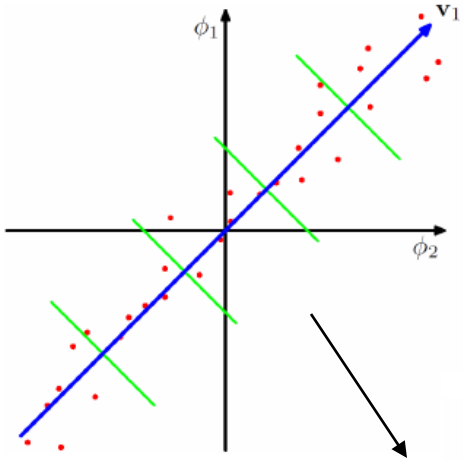
KPCA Recap

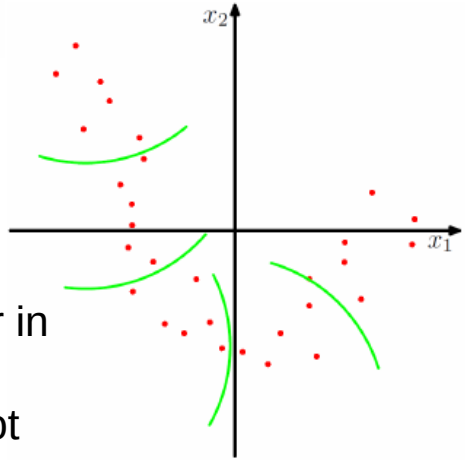
- The eigenvectors now reside in the high dimensional space H

$$\mathbf{v}^{(k)} = \sum_{i=1}^{N} \alpha_i^{(k)} \Phi\left(\mathbf{x_i}\right)$$

- This implies that kernel PCA can be used to feature extraction but CANNOT be used (directly) for reconstruction

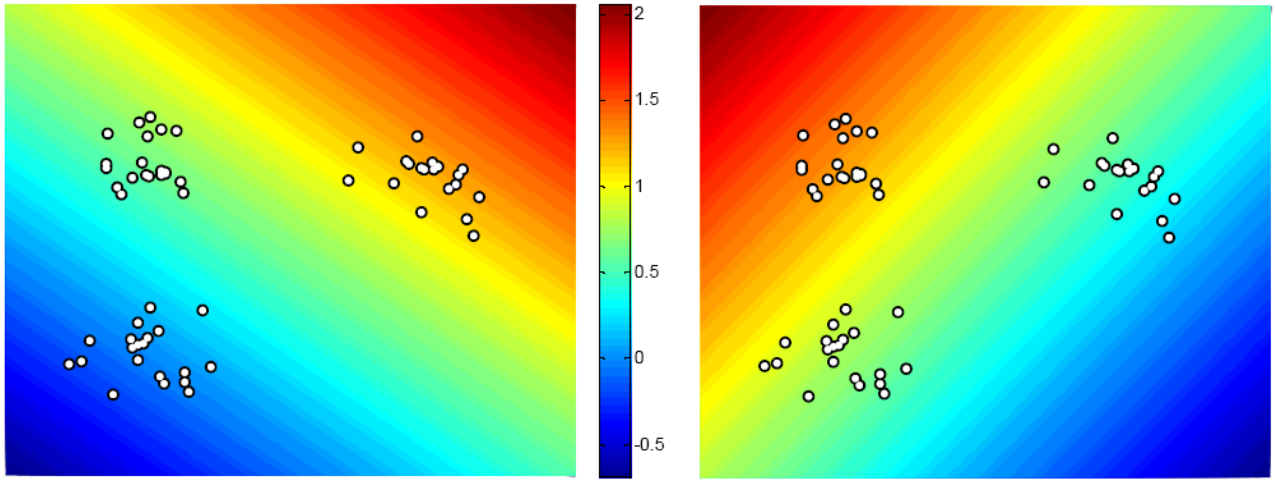


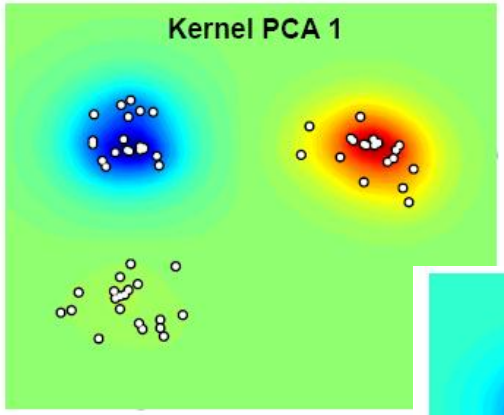


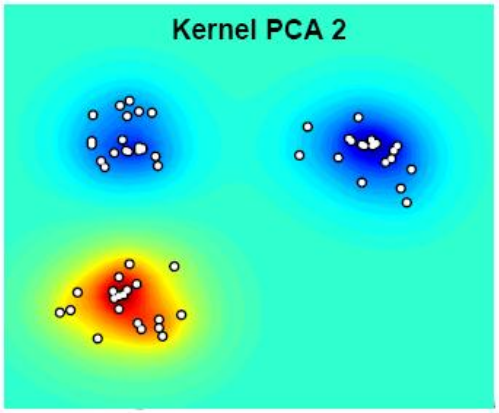


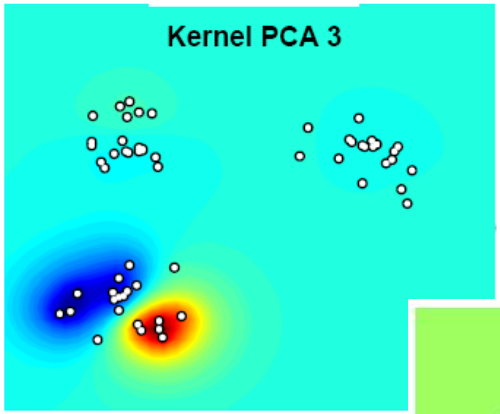
The subspace achieved by KPCA is nonlinear in the input space, which allows KPCA to be applied to complex real-world data that are not linearly separable.

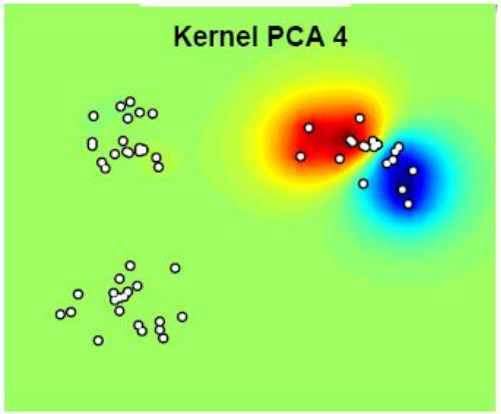
PCA vs Kernel PCA



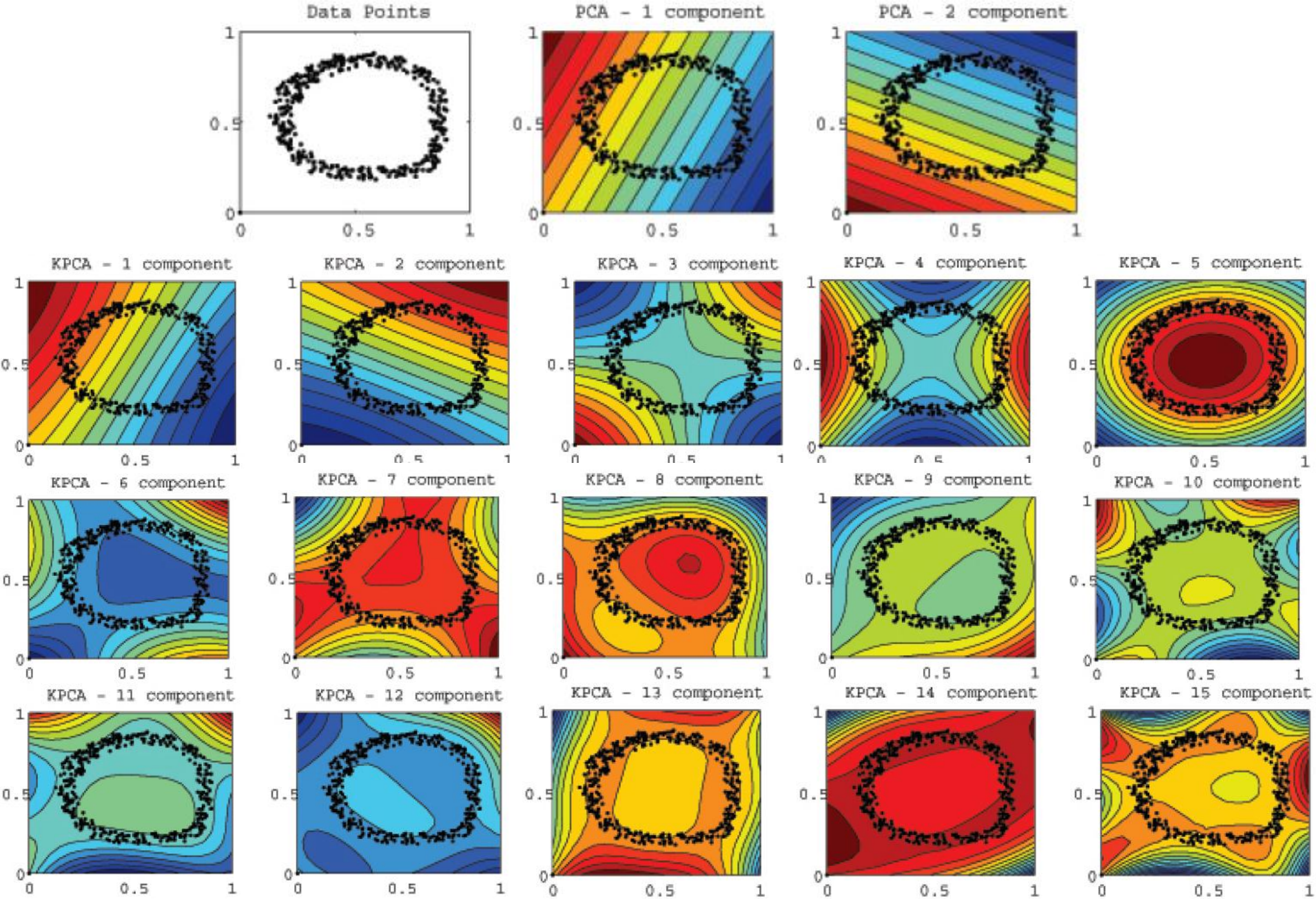








Courtesy of R. Gutierrez


Data Points
PCA - 1 component
PCA - 2 component
KPCA - 1 component
KPCA - 2 component
KPCA - 3 component
KPCA - 4 component
KPCA - 5 component
KPCA - 6 component
KPCA - 7 component
KPCA - 8 component
KPCA - 9 component
KPCA - 10 component
KPCA - 11 component
KPCA - 12 component
KPCA - 13 component
KPCA - 14 component
KPCA - 15 component

Applications Of KPCA

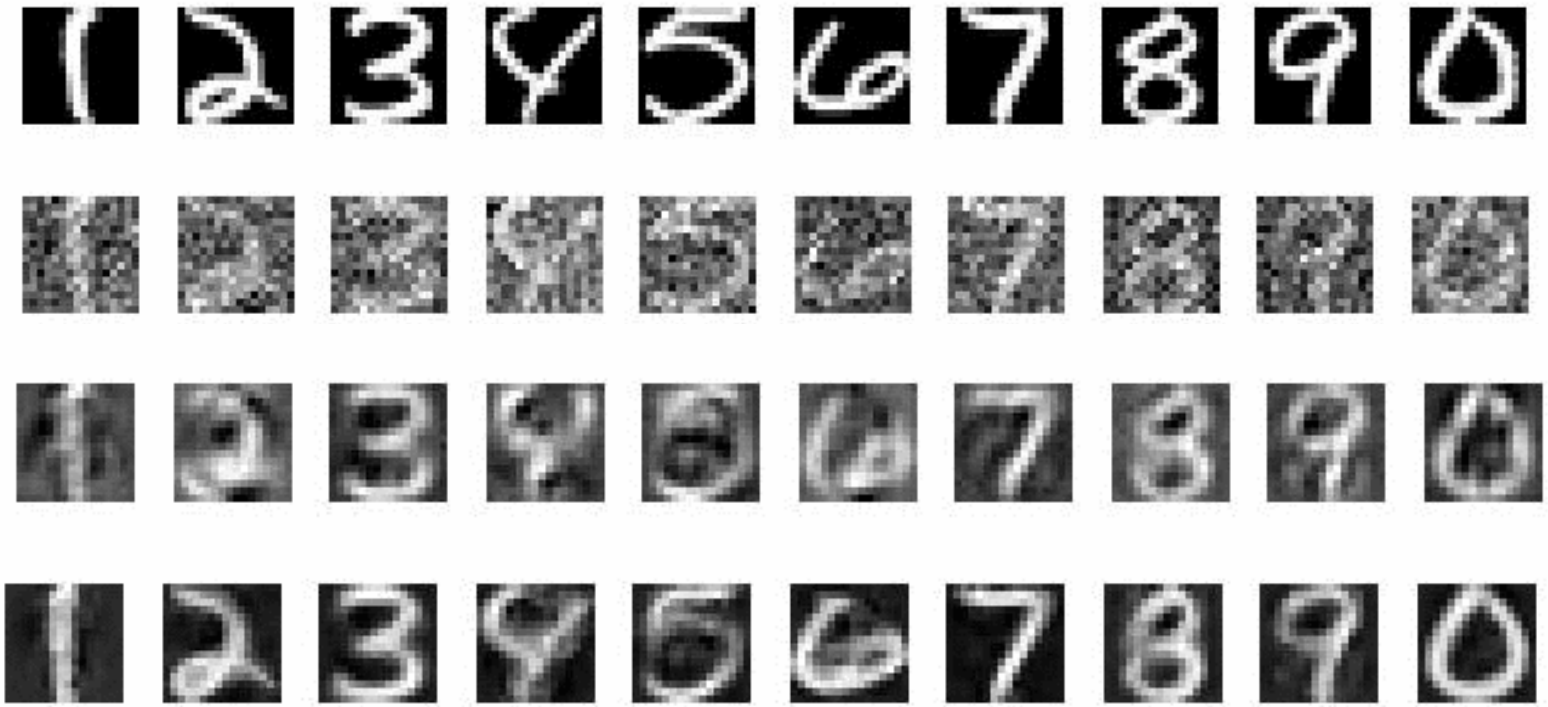
Denoising of the USPS hand-written numerals corrupted by Gaussian noise

Ground truth

Noisy images

Linear PCA

Kernel PCA



References

- B. Schökopf and A. Smola, “Learning with Kernels”
- J. Shawe-Taylor and N. Cristianini, “Kernel Methods for Pattern Analysis”

Recap

- Linear vs Non-Linear Methods
- Kernel Mapping
- The Kernel Trick
- Common Kernels
- Kernel PCA (KPCA)