NONLINEAR COMPONENT ANALYSIS AS A KERNEL EIGENVALUE PROBLEM

Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller

Karthik | Naman | Shubham | Zhenye | Ziyu

Department of Industrial and Enterprise Systems Engineering

Overview

- Introduction and Motivation
  - Review of Principal Component Analysis
  - Problem of PCA
  - Strategy Implementation
  - Computational Hurdles
  - Introduction of Kernels
- Technical Background
  - Kernel Methods
- Summary of Main Results
  - Pseudocodes and Algorithm
  - Experimental Results of the Paper
- Application Examples
  - Toy Example
  - IRIS Clustering
  - USPS Classification
- Summary and Connection to the Course
- References

INTRODUCTION AND MOTIVATION

Review : Principal Component Analysis

Motivation:

- Reduce the dimensions of the dataset with minimal loss of information.

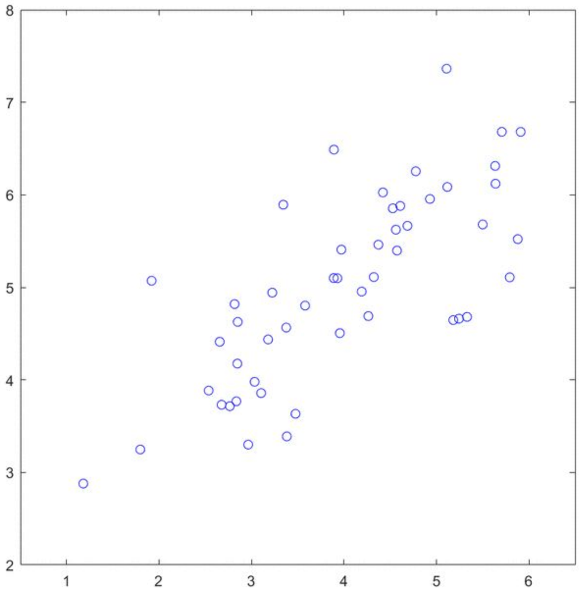
Definition:

- PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

How to perform linear PCA?

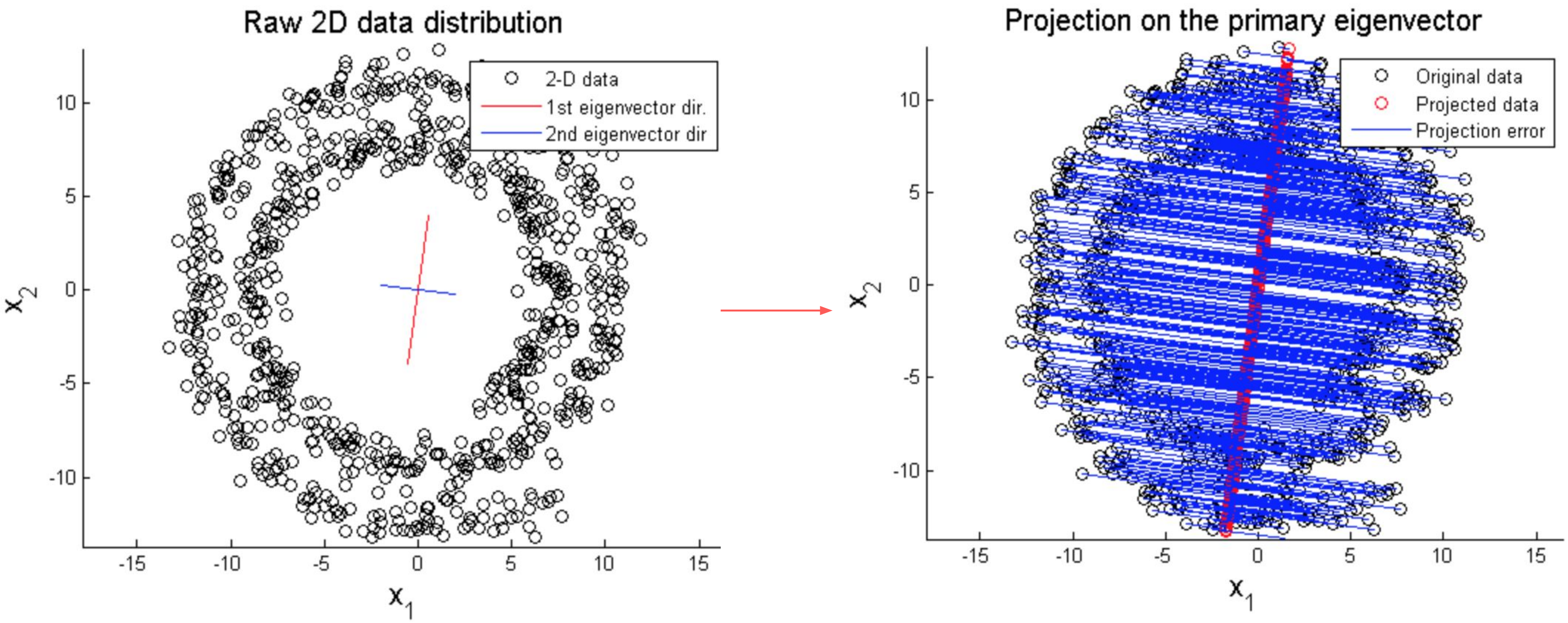
De-mean data → Scale data → Compute covariance matrix → Perform eigenvalue decomposition → Sort eigenvalues & get sorted eigenvectors

Principal Component Analysis in Action:



- Determining the axis (component) of maximum variance.
- Finding all such orthogonal component.
- Projecting the data on those components.

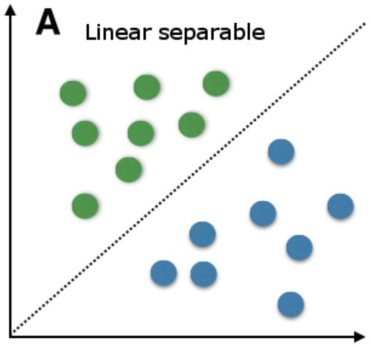
Principal Component Analysis in Action:

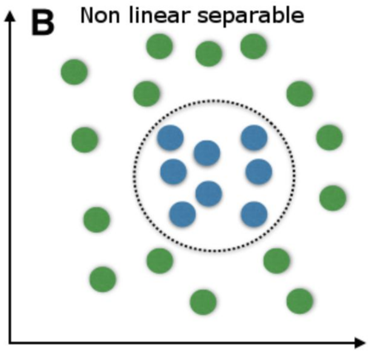


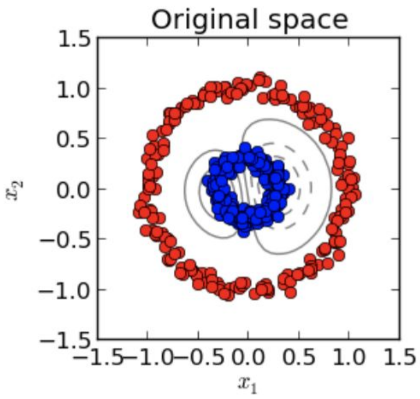
- **Problem: Determining the axis (component) of maximum variance.**

Principal Component Analysis in Action:

- **Other examples:**
  - Facial images with emotional expressions
  - Images of an object of which orientation is variable
  - Data that can't be separated by linear boundaries







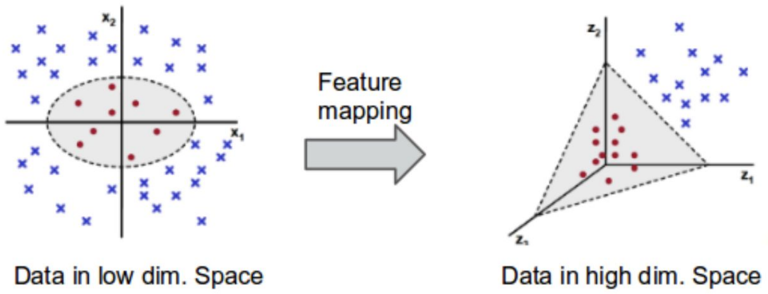
Problem of PCA

Problem Statement:

- Unable to find components that represents nonlinear data effectively.
- Information loss with projected data.

Strategy to tackle this problem:

- Map data to higher dimension.
    - **Assumption:** The data will be linearly distributed in higher dimensions.
- Perform PCA in that space.
- Project datapoint on that PC's



Strategy Implementation

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

- **F** - Feature Space
- **Φ** - Transforming function
- **M** - Total number of observations
- **N** - Total number of features
- **x** - Original data with M observations and N features

$$\begin{array}{c c} & \begin{array}{cccc} F1 & F2 & \dots & FN \end{array} \\ \begin{array}{c} \text{Obs1} \\ \text{Obs2} \\ \\ \text{ObsM} \end{array} & \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & x_{2N} \\ & & & \\ x_{M1} & x_{M2} & \dots & x_{MN} \end{bmatrix} \end{array}$$

Transform data into F space

↓

Centralize the data in F space

↓

Perform PCA
(same steps as linear PCA)

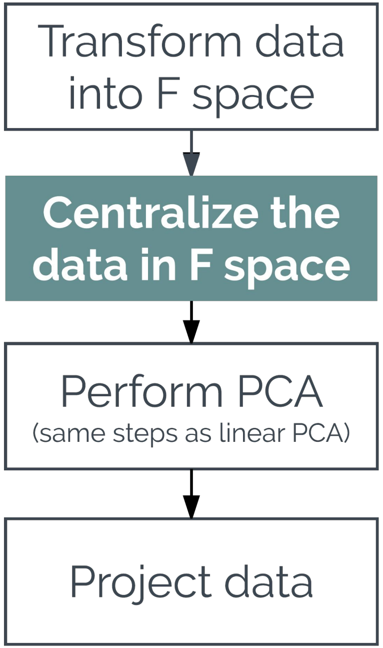
↓

Project data

Strategy Implementation

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

$$\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0.$$



Strategy Implementation

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

$$\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0.$$

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T$$

$$\lambda \mathbf{V} = \bar{C}\mathbf{V}$$

Transform data into F space

↓

Centralize the data in F space

↓

Perform PCA
(same steps as linear PCA)

↓

Project data

Strategy Implementation

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

$$\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0.$$

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T$$

$$\lambda \mathbf{V} = \bar{C}\mathbf{V}$$

$$(\Phi(\mathbf{x}_k) \cdot \mathbf{V})$$

Transform data into F space

↓

Centralize the data in F space

↓

Perform PCA
(same steps as linear PCA)

↓

Project data

Computational hurdles

- **Problem:**
  - We want to take the advantage of mapping into high-dimensional space.
  - The mapping, however, can be arbitrary, with a very high or infinite dimensionality.
  - Computing the mapping of each data point to that space will be computational expensive.

$$\bar{C} = \frac{1}{M}\sum_{j=1}^{M}\Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T$$

$$\lambda\mathbf{V} = \bar{C}\mathbf{V}$$

Introduction of Kernels

One method to solve that computational problem is to use 'KERNELS'.

Definition:

- Kernels are functions that perform dot product in transformed space.

$$\kappa(x, y) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$$

- Some examples for kernels:

Kernel Name	Function Expression
Linear	$K(x, y) = x^T y$
Polynomial	$K(x,y) = \left(1 + \frac{x^T y}{\sigma^2}\right)^d$
RBF	$K(x,y) = \exp\left\{- \frac{\|x-y\|^2}{\sigma^2}\right\}$

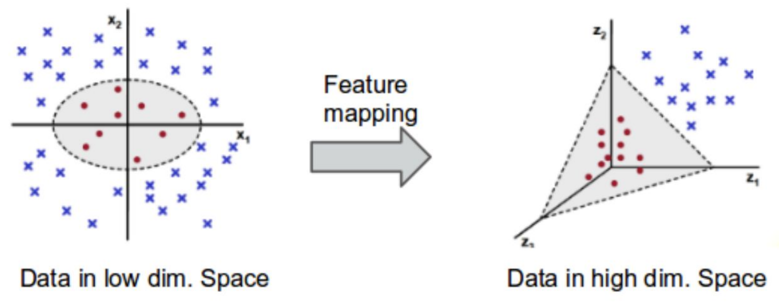
Introduction of Kernels

Why 'KERNELS' are computationally efficient?

Reason:

- computing dot product in transformed space, without explicitly carrying out the entire data transformation..

Example:



$$\Phi = \mathbf{R}^2 \mapsto \mathbf{R}^3,\ (x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\begin{aligned}\phi(x)^T\phi(z) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T(z_1^2, \sqrt{2}z_1z_2, z_2^2)\\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2\\ &= (x_1z_1 + x_2z_2)^2\\ &= (x^Tz)^2\\ &= \mathcal{K}(x, z)\end{aligned}$$

$$\kappa(x, y) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$$

TECHNICAL BACKGROUND

Algebraic Manipulations

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V})$$

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

$$\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0.$$

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T$$

$$\boxed{\lambda \mathbf{V} = \bar{C}\mathbf{V}}$$

$$(\Phi(\mathbf{x}_k) \cdot \mathbf{V})$$

Algebraic Manipulations

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V})$$

$$\downarrow$$

$$\mathbf{V} = \sum_{j=1}^{M} \alpha_i \Phi(\mathbf{x}_i)$$

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

$$\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0.$$

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T$$

$$\boxed{\lambda \mathbf{V} = \bar{C}\mathbf{V}}$$

$$(\Phi(\mathbf{x}_k) \cdot \mathbf{V})$$

Algebraic Manipulations

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V})$$

$$\downarrow$$

$$\mathbf{V} = \sum_{j=1}^{M} \alpha_i \Phi(\mathbf{x}_i)$$

$$\downarrow$$

$$\lambda \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_i))$$

$$= \frac{1}{M} \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \sum_{j=1}^{M} \Phi(\mathbf{x}_j))(\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i))$$

$$\Phi : \mathbf{R}^N \to F, \mathbf{x} \mapsto \mathbf{X}$$

$$\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0.$$

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x}_j) \Phi(\mathbf{x}_j)^T$$

$$\boxed{\lambda \mathbf{V} = \bar{C}\mathbf{V}}$$

$$(\Phi(\mathbf{x}_k) \cdot \mathbf{V})$$

Kernel Method for PCA

$$\boxed{\lambda \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_i)) = \frac{1}{M} \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \sum_{j=1}^{M} \Phi(\mathbf{x}_j))(\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i))}$$

$$\mathbf{K}_{ij} = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \longrightarrow M\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}$$

Kernel Method for PCA

$$\lambda \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_i)) = \frac{1}{M} \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \sum_{j=1}^{M} \Phi(\mathbf{x}_j))(\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i))$$

$$\mathbf{K}_{ij} = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \longrightarrow M\lambda \boldsymbol{\alpha} = K\boldsymbol{\alpha}$$

<u>Note:</u>

The equations looks like eigenvalue decomposition of matrix **K**

Projection Using Kernel Method

$$(\Phi(\mathbf{x}_k) \cdot \mathbf{V})$$

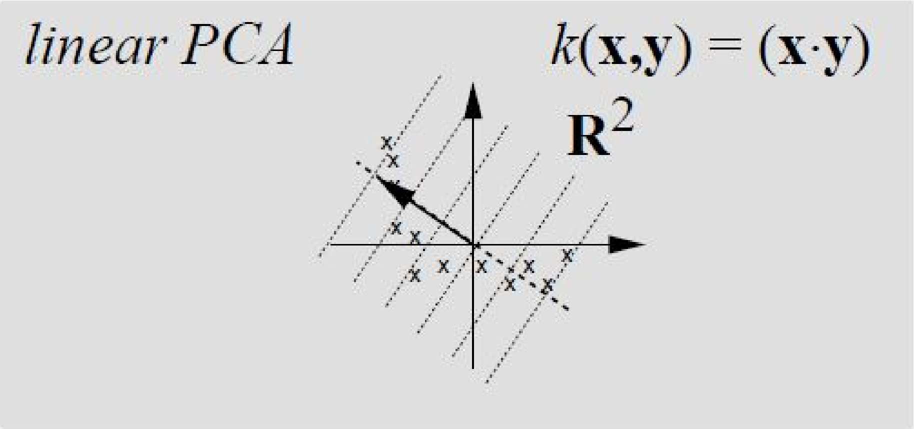
$$\mathbf{V} = \sum_{j=1}^{M} \alpha_i \Phi(\mathbf{x}_i)$$

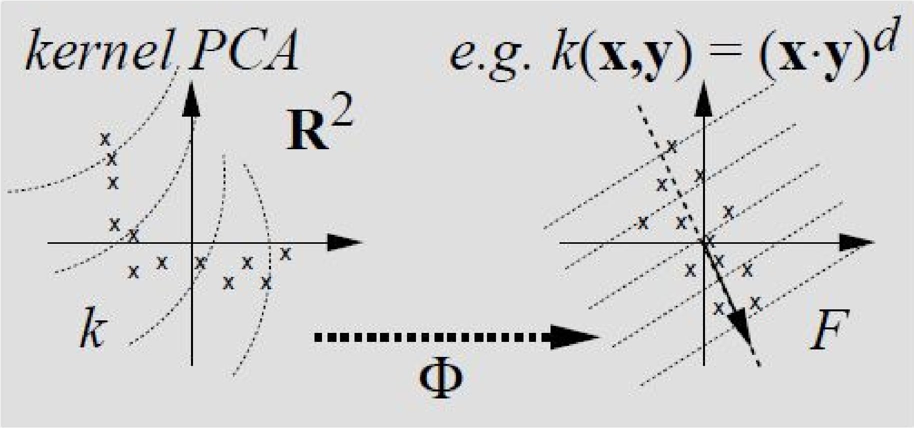
$$\sum_{i=1}^{M} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}))$$

$$\kappa(x, y) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$$

$$(\mathbf{V}^n \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^n \kappa(\mathbf{x}_i, \mathbf{x})$$

Visual Representation : KPCA





KPCA steps in a nutshell

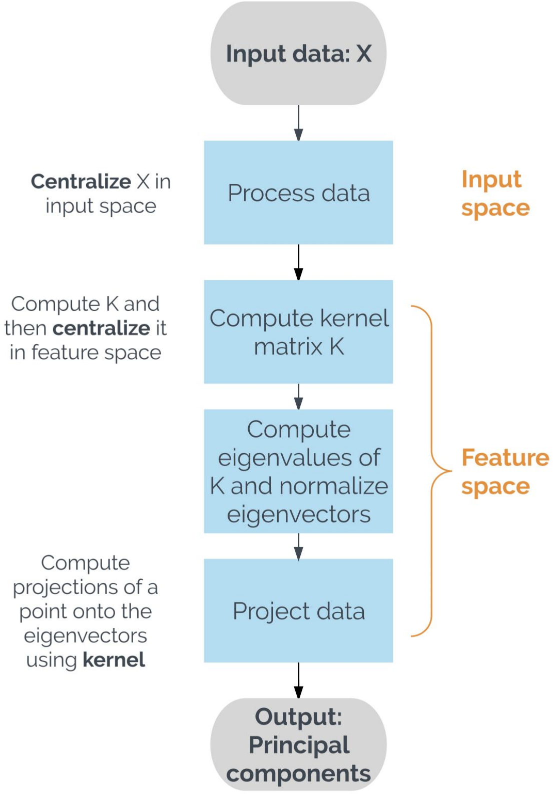
The following steps were necessary to compute the principal components:

1. Compute the kernel matrix K,
2. Compute its eigenvectors and normalize them in F, and
3. Compute projections of a test point onto the eigenvectors.

SUMMARY OF MAIN RESULTS

Kernel PCA: Pseudocode

- Loading Test Data
- Centering Test data
- Creating Kernel K matrix
- Centering of Kernel K matrix in F space
- Eigenvalue Decomposition of K centered Matrix
- Sorting Eigenvalues in descending order.
- Selecting the significant eigenvectors corresponding to these eigenvalues.
- Normalizing all significant sorted eigenvectors of K
- Projecting data in the principal component coordinate system



Algorithm For Kernel PCA

Algorithm 1 Kernel PCA Algorithm

procedure K - PCA(X)

Given Input: $X_{N\times M} \leftarrow [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M]$

Centralize : $X_{centered} \leftarrow X_{N\times M}$

Kernel Matrix : $K_{M\times M} : k_{ij} \leftarrow k(\mathbf{x}_i, \mathbf{x}_j)$

Centralization in F space :

$K : K \leftarrow K - I_M K/M - K I_M/M + I_M K I_M/M^2$

Extracting eigenvectors : $M\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}$

Normalization :

$\boldsymbol{\alpha} \leftarrow \frac{\boldsymbol{\alpha}}{mod(\boldsymbol{\alpha})\sqrt{M\lambda}}$

loop: $i \leftarrow 1 : p$

$$P_i(x) = \sum_{i=1}^{M} \alpha_{ij}\kappa(\mathbf{x}_i, \mathbf{x})$$

goto *top*.

COMPUTATIONAL COMPLEXITY

- A fifth-order polynomial kernel on a 256-dimensional input space yields a 10^{10} dimensional feature space

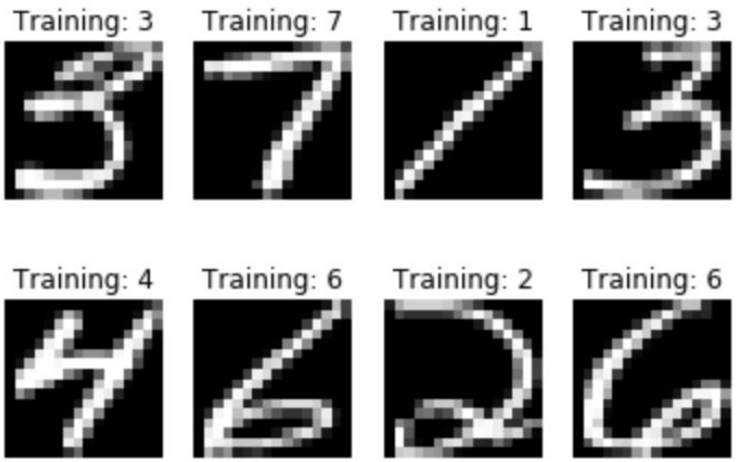
- We have to evaluate the kernel function M times for each extracted principal component ,rather than just evaluating one dot product as for a linear PCA.

$$(\mathbf{V}^n \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^n \kappa(\mathbf{x}_i, \mathbf{x})$$

- Finally, although kernel principal component extraction is computationally more expensive than its linear counterpart, this additional investment can pay back afterward.

USPS Handwriting Dataset

The dataset refers to numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. The images have been de-slanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990).



Scan me!

LINK TO USPS REPO :
https://cs.nyu.edu/~roweis/data.html

Experimental Results of Article

- Nonlinear PCs afforded better recognition rates than corresponding numbers of linear PCs.

- Performance for nonlinear components can be improved by using more components than is possible in the linear case.

Test Error Rates on the USPS Handwritten Digit Database

	Test Error Rate for Degree						
Number of components	1	2	3	4	5	6	7
32	9.6	8.8	8.1	8.5	9.1	9.3	10.8
64	8.8	7.3	6.8	6.7	6.7	7.2	7.5
128	8.6	5.8	5.9	6.1	5.8	6.0	6.8
256	8.7	5.5	5.3	5.2	5.2	5.4	5.4
512	N.A.	4.9	4.6	4.4	5.1	4.6	4.9
1024	N.A.	4.9	4.3	4.4	4.6	4.8	4.6
2048	N.A.	4.9	4.2	4.1	4.0	4.3	4.4

APPLICATION EXAMPLES

EXAMPLE APPLICATIONS

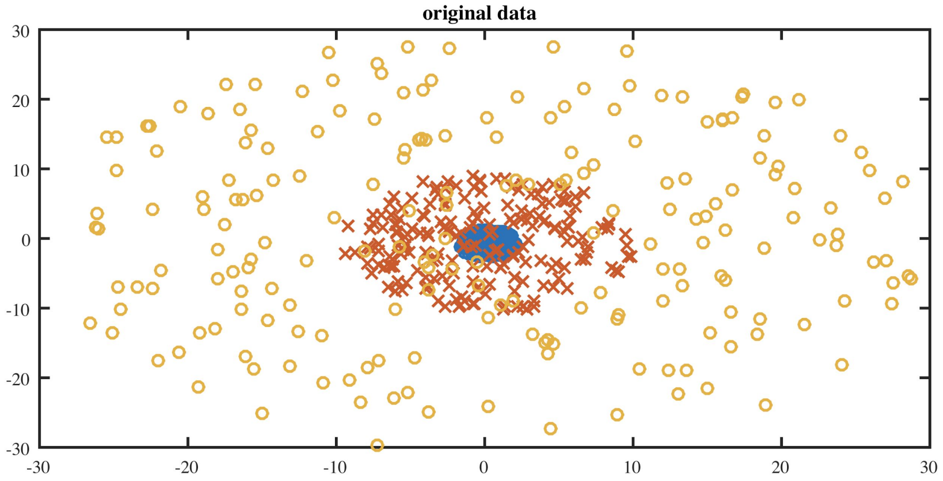
Scan me!

1. TOY EXAMPLE
2. IRIS Clustering
3. USPS Classification

LINK TO OUR GITHUB REPO :
https://github.com/Zhenye-Na/npca

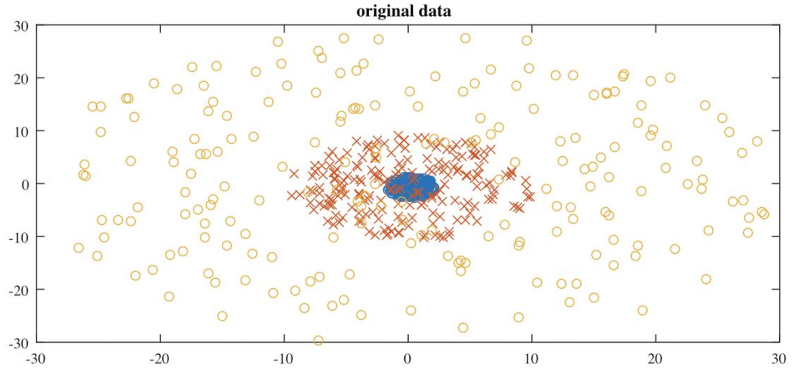
Toy Example

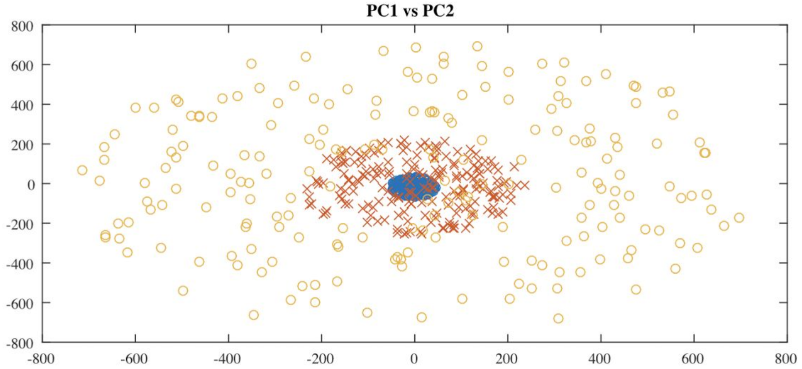
- The idea is to test kernels before implementing it on larger datasets.
- Created our own dataset
- Programming Language Used: MATLAB



Toy Example

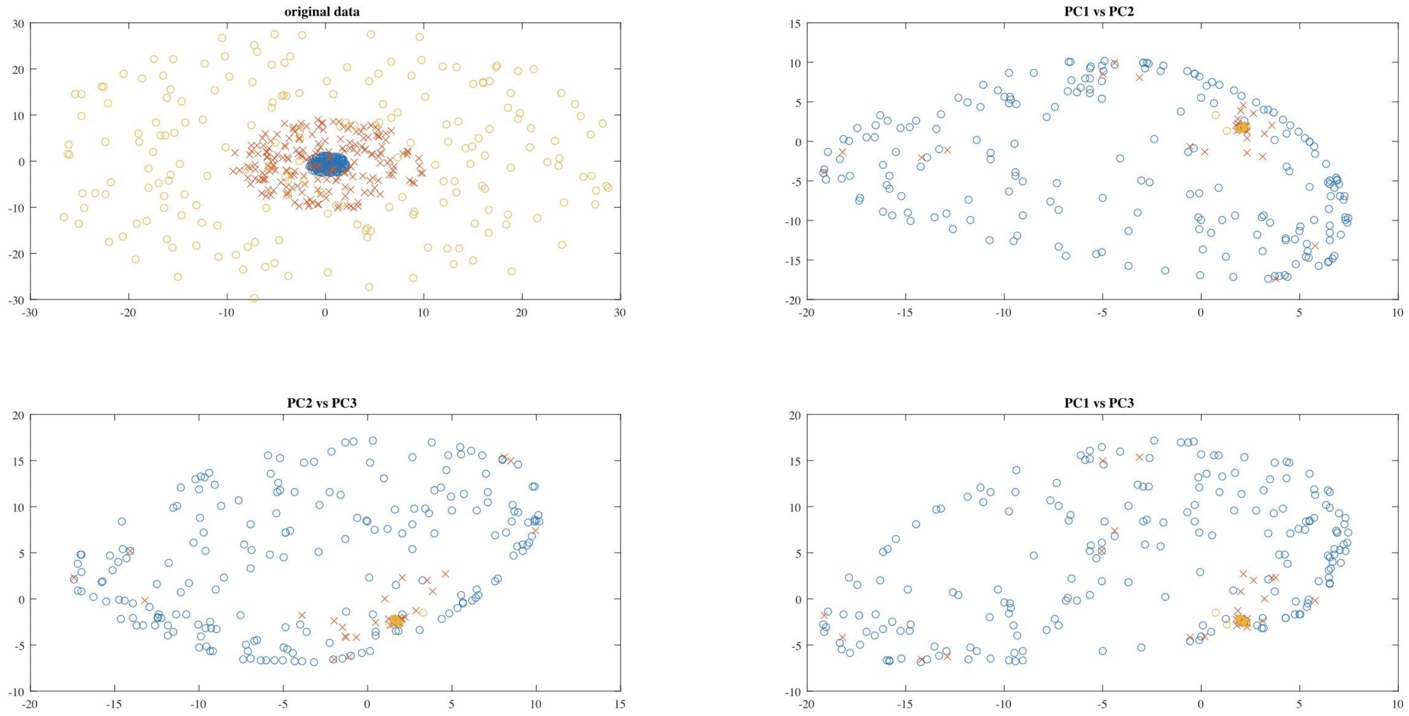
Case 1: Linear Kernel is used





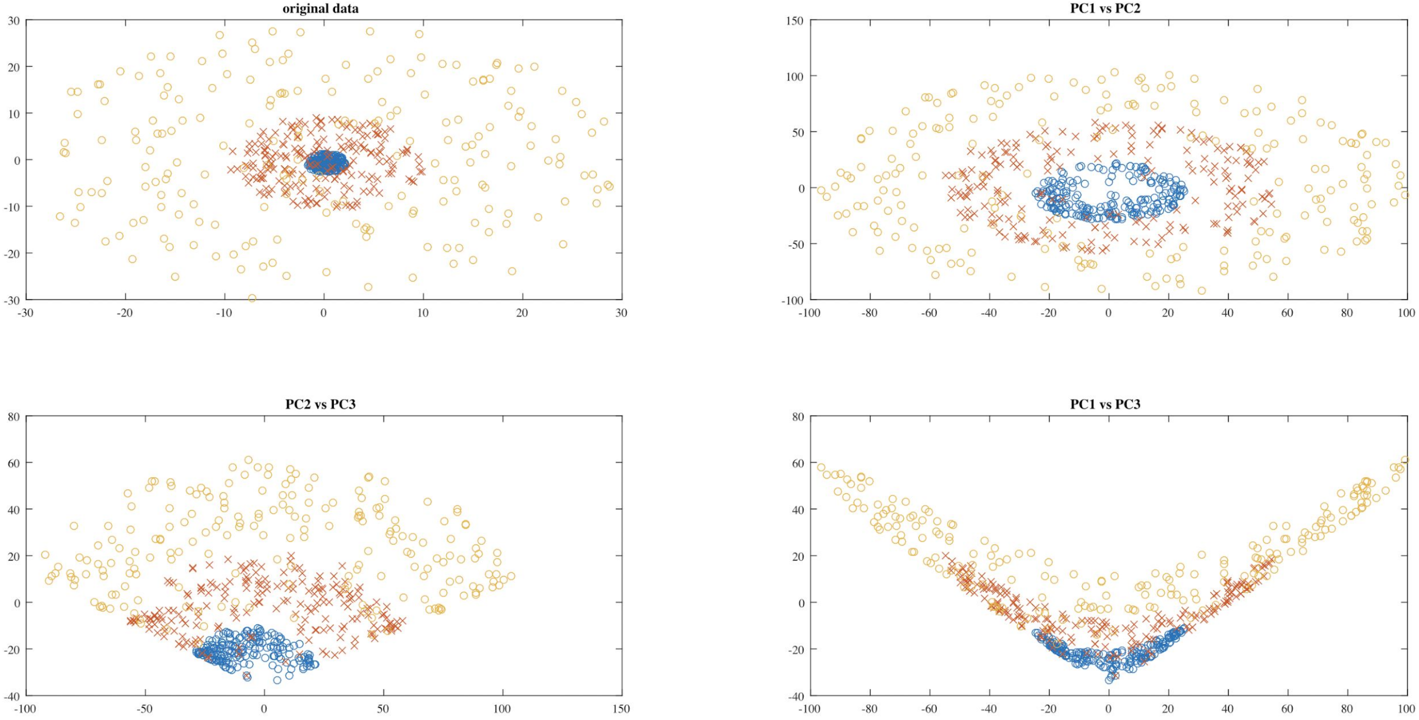
Toy Example

Case 2: Gaussian kernel is used



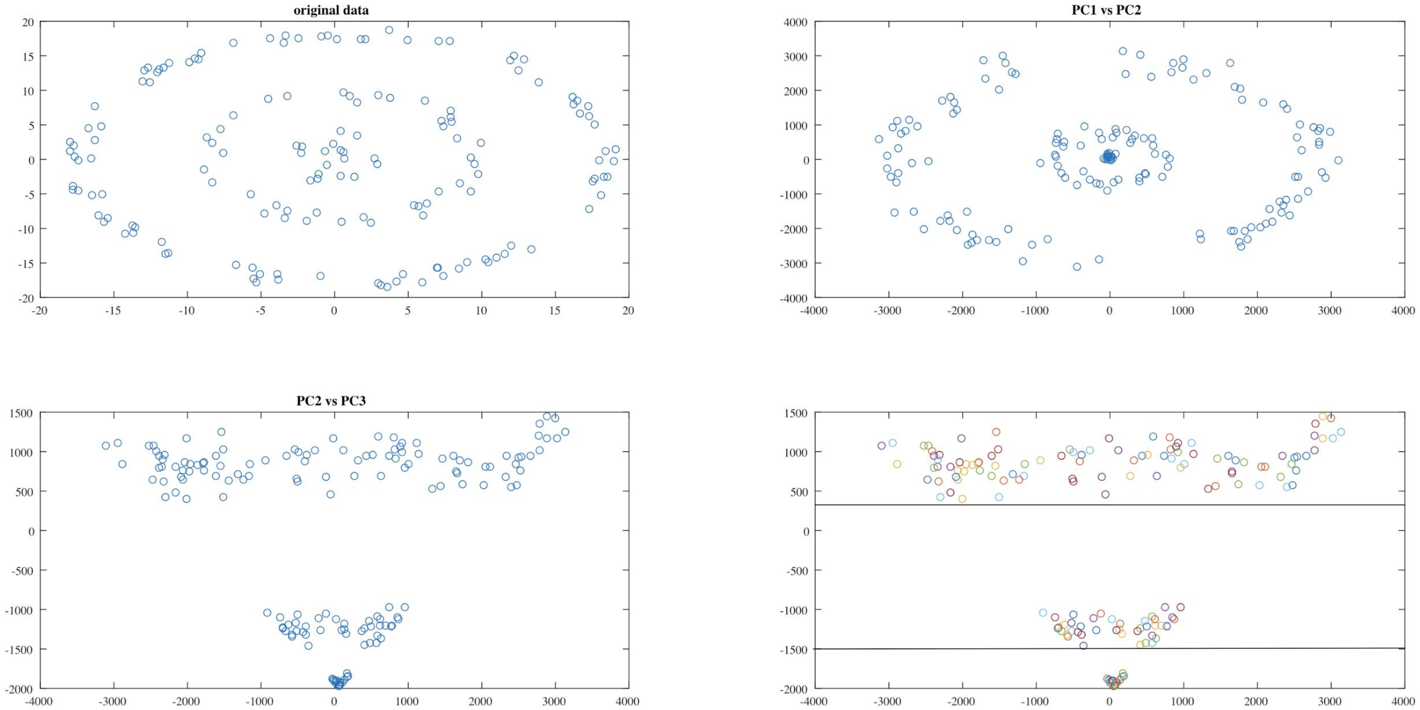
Toy Example

Case 3: Polynomial (Degree = 0.5) is used



Toy Example

Case 4: Polynomial (Degree = 2) is used



IRIS Clustering

Scan me!

The idea is to figure out if we could cluster out the iris flower data set and find out more inherent clusters.

Programming Language Used:
MATLAB

Repository : UCI Machine Learning Repository

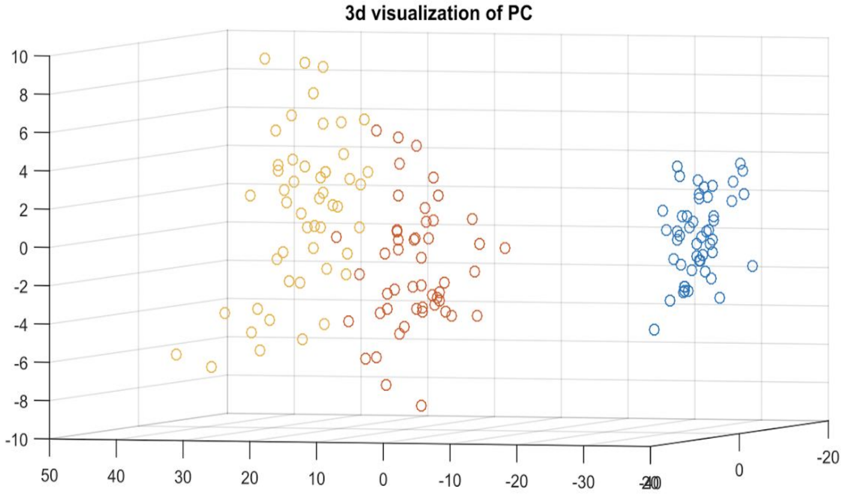
LINK TO UCI REPO : https://archive.ics.uci.edu/ml/datasets/iris

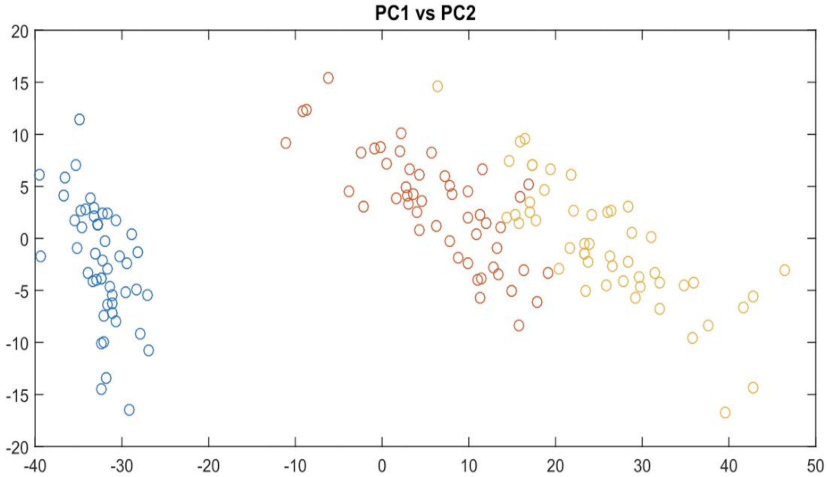
IRIS DATASET

- Same as in computational assignment
- Dataset was obtained from UCI database. Three flower species were considered and there are four features.
- Observations were taken in rows and features in columns.
- Only two visible clusters were obtained from Linear PCA.
- We expected to obtain more information through Kernel PCA, but got only two clusters although there are three species of flowers.

IRIS Clustering

Case 1: Linear Kernel is used

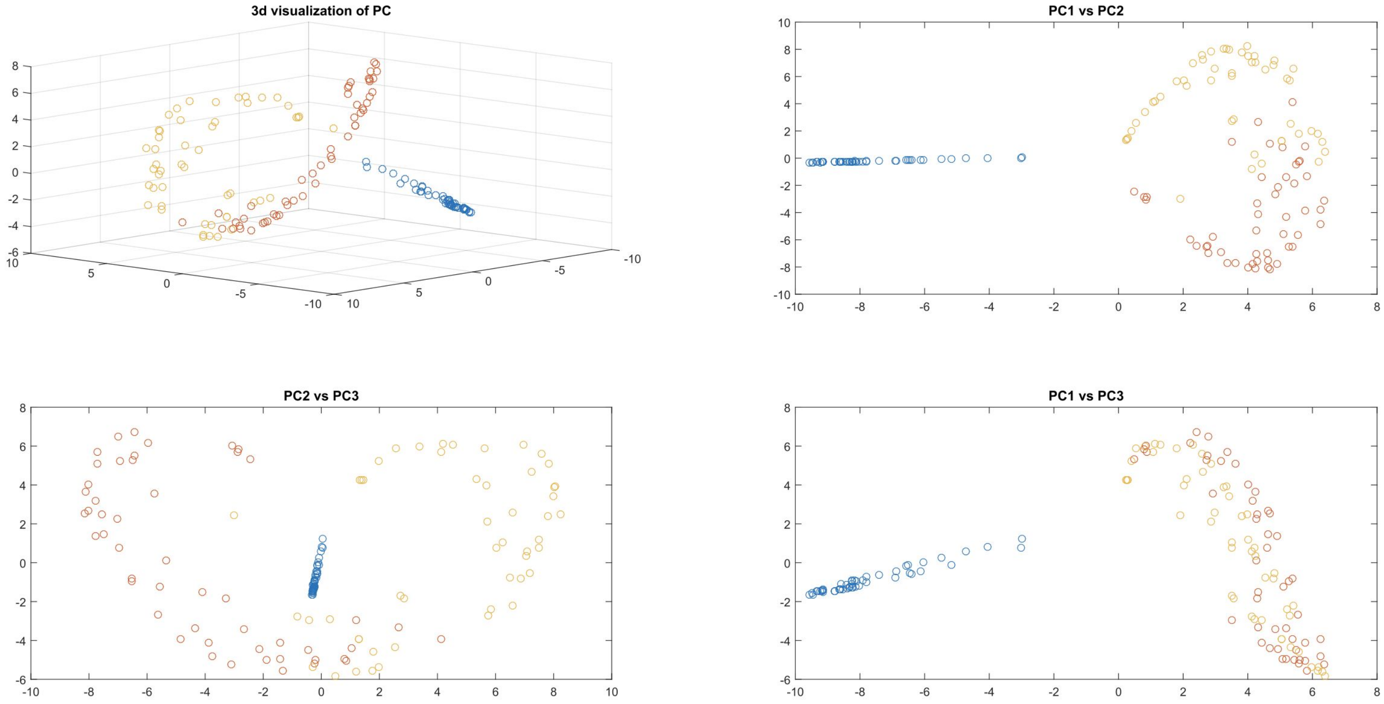




Results: No apparent data separation is observed

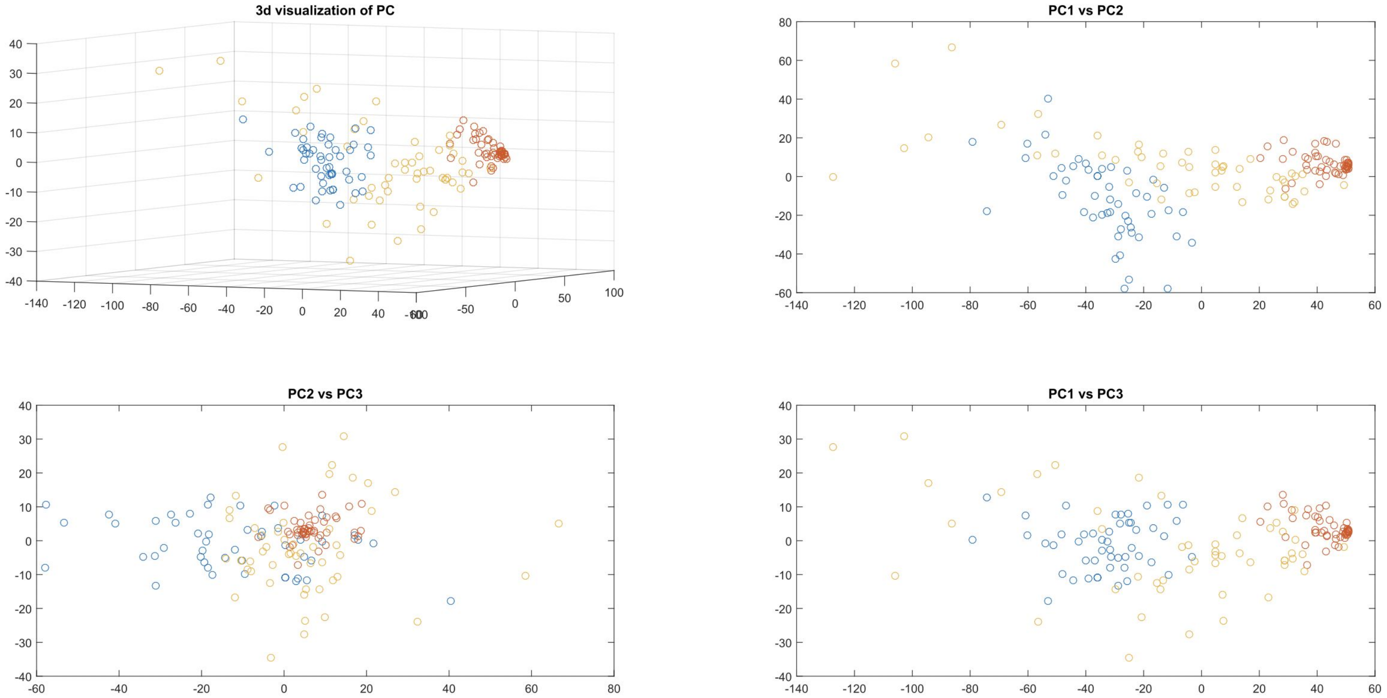
IRIS Clustering

Case 2: Gaussian Kernel is used



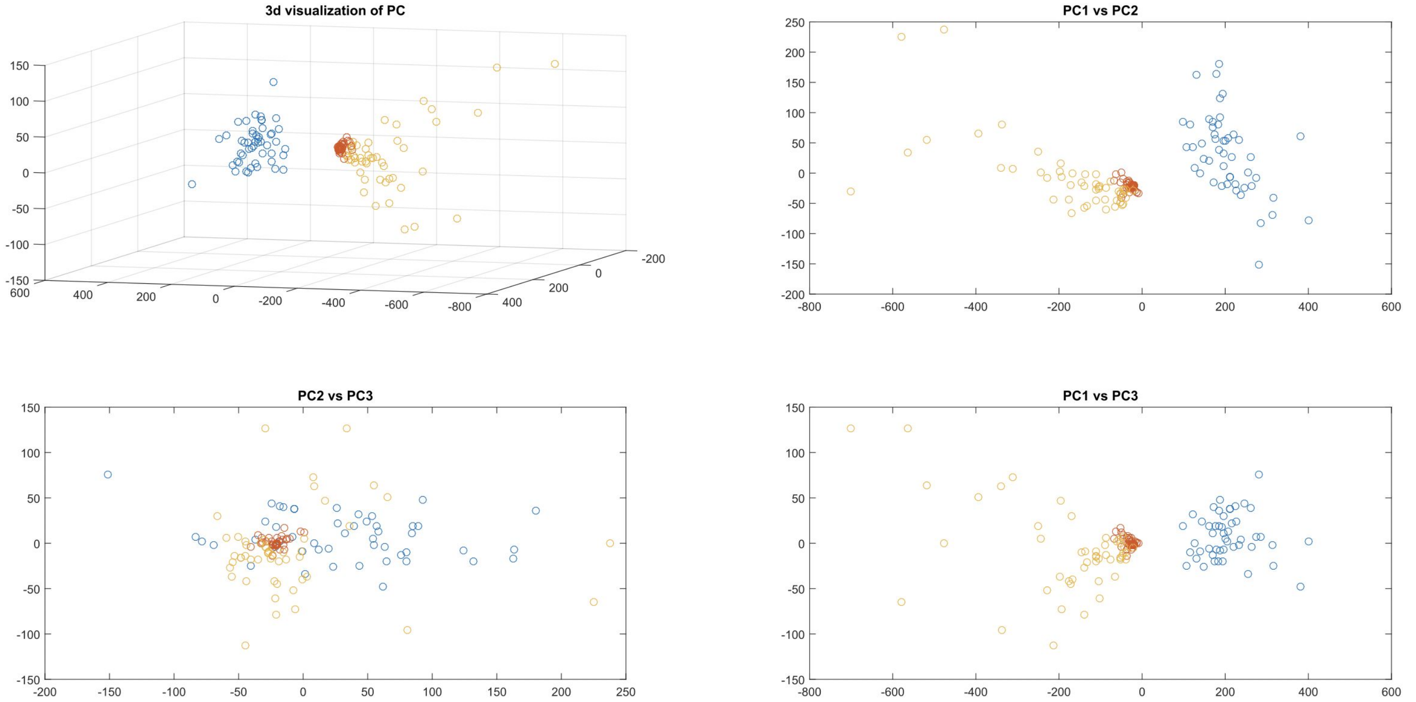
IRIS Clustering

Case 3: Polynomial (Degree = 2) Kernel is used



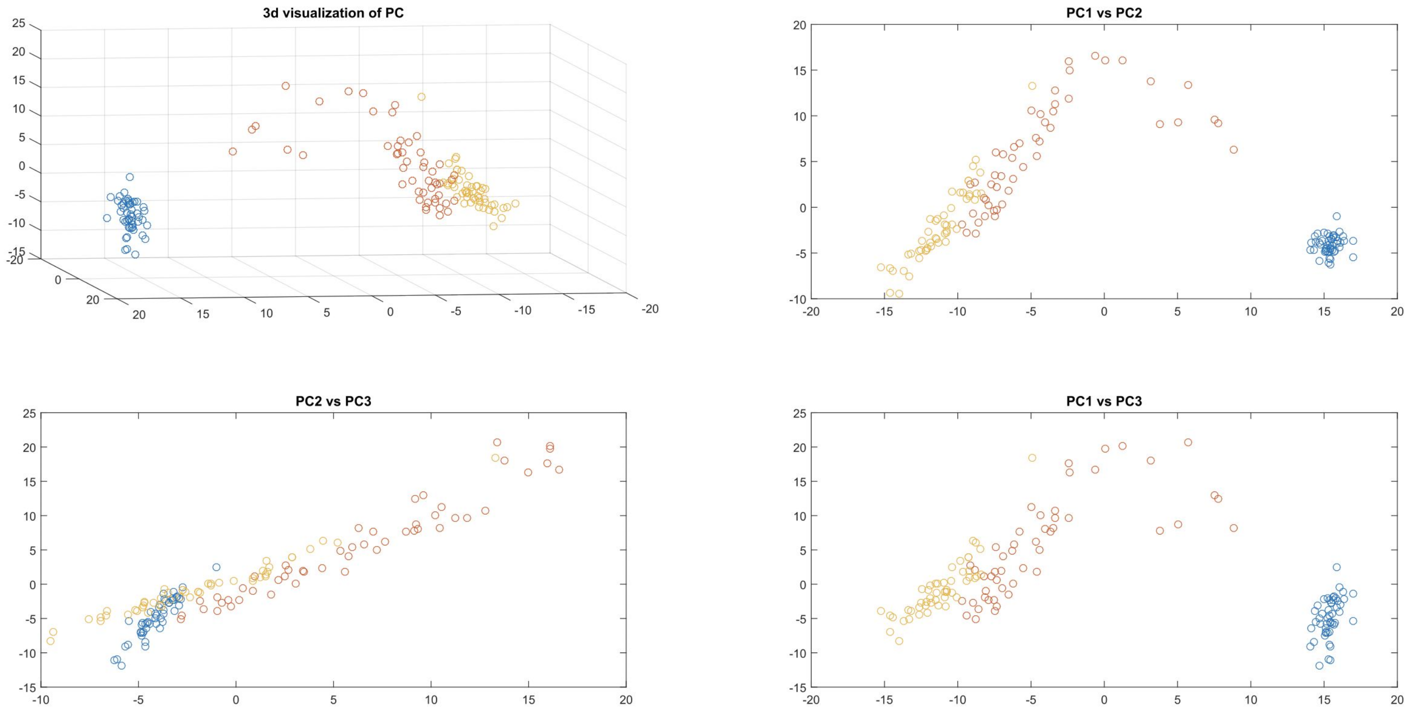
IRIS Clustering

Case 4: Polynomial (Degree = 3) Kernel is used



IRIS Clustering

Case 5: Polynomial(Degree = 0.5) is used

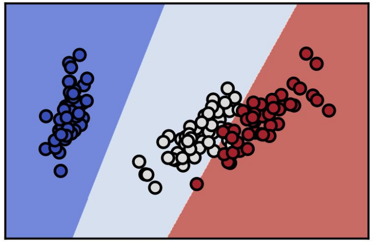


IRIS Classification

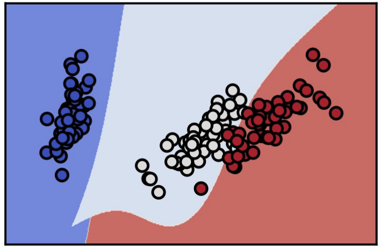
PCA → SVM

Perform Kernel PCA with RBF on original data and then perform SVM. The scores in the chart below are the mean accuracy on the given test data and labels.

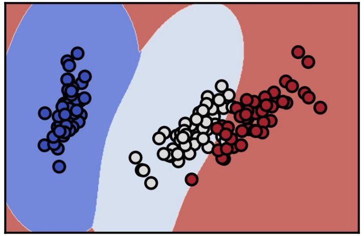
SVC with linear kernel

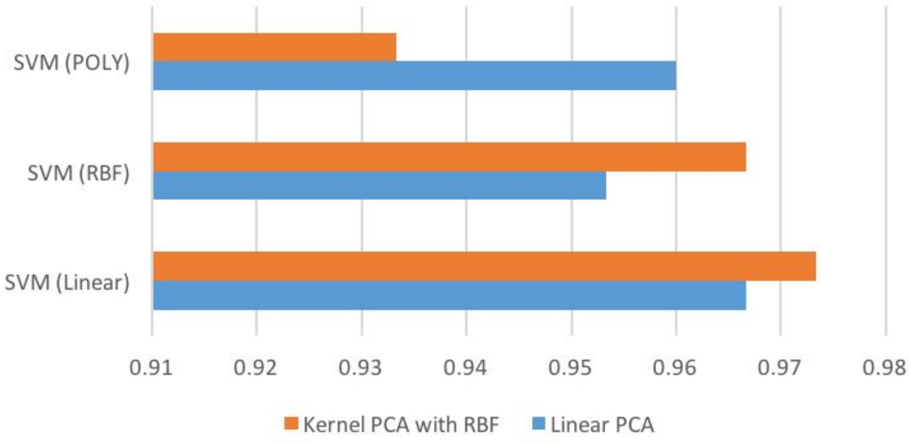


SVC with polynomial kernel



SVC with RBF kernel

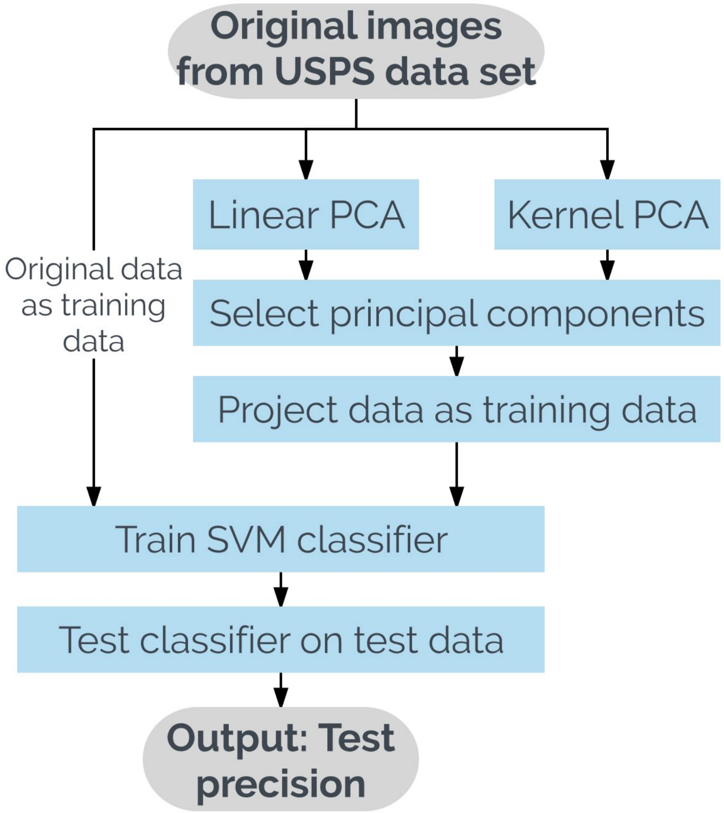




USPS HANDWRITING RECOGNITION

- ➔ USPS Dataset contains numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service.
- ➔ Feature extraction is done via PCA and Kernel PCA with polynomial kernel.
- ➔ Training set: 8000 x 256; Test set: 3000 x 256.
- ➔ Applied to a SVM (with Linear Kernel) classifier to train and test on the splitted USPS dataset.
- ➔ We expected to see higher accuracy given by Kernel PCA than Linear PCA during the classification.

Experiments Setup



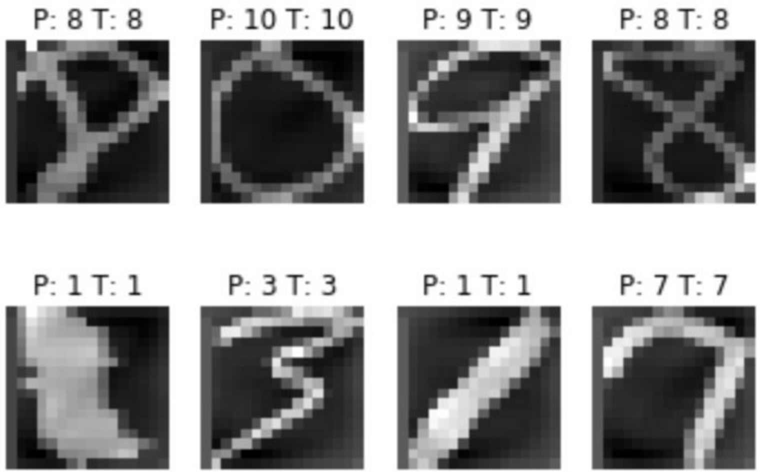
Select the smallest number of principal components which can contain **99%** of variance

Data Preprocessing - Feature Scaling

Standardize features by removing the mean and scaling to unit variance.

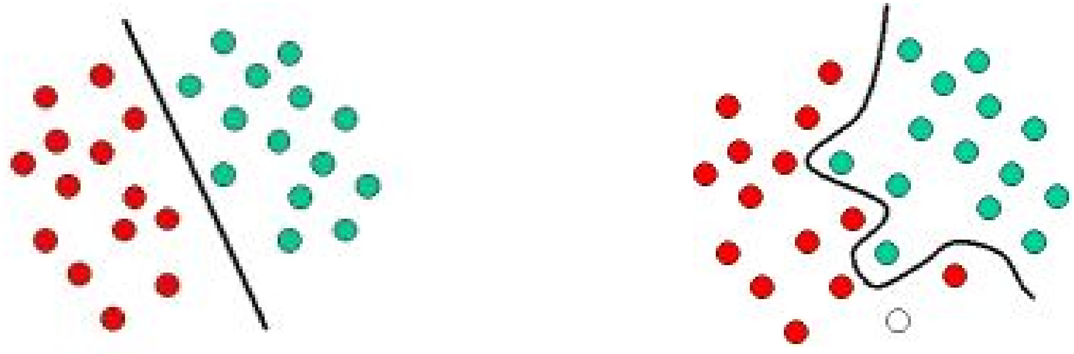
Before:

After:



SVM - Introductory Overview

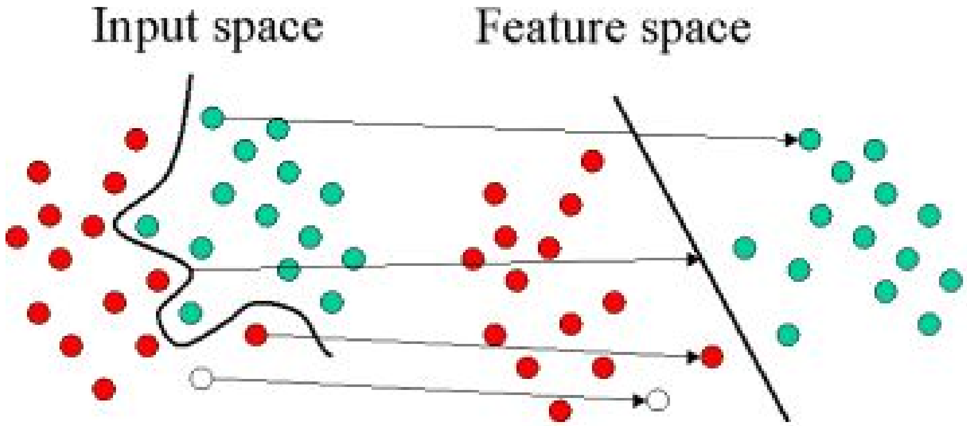
Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. Any new object falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).



https://www.youtube.com/watch?v=_PwhiWxHK8o&list=RDQM83CF7-lddZA

SVM - Introductory Overview

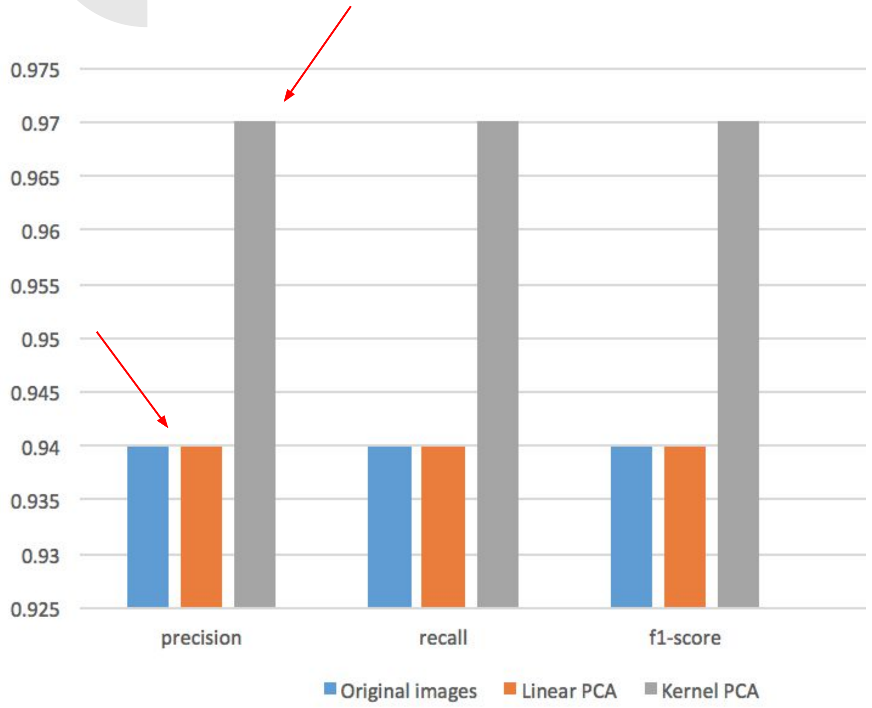
Here we see the original objects (left side of the schematic) mapped, i.e., rearranged, using kernels. Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all we have to do is to find an optimal line that can separate the GREEN and the RED objects.

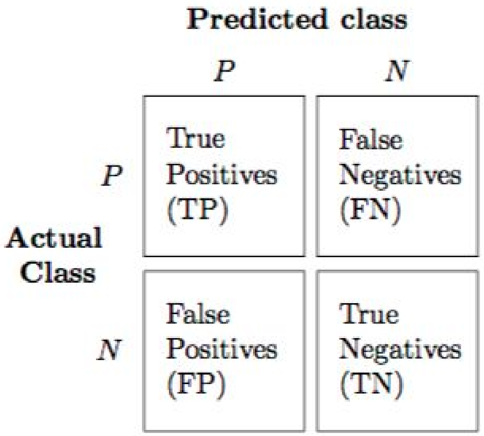


USPS Data Classification

	Original Image	Features PCA	Features KPCA (deg 2)	Features KPCA (deg 3)
SVM (linear)	1	2	3	4
SVM (deg 2)	5	6	7	8
SVM (deg 3)	9	10	11	12

SVM results summary



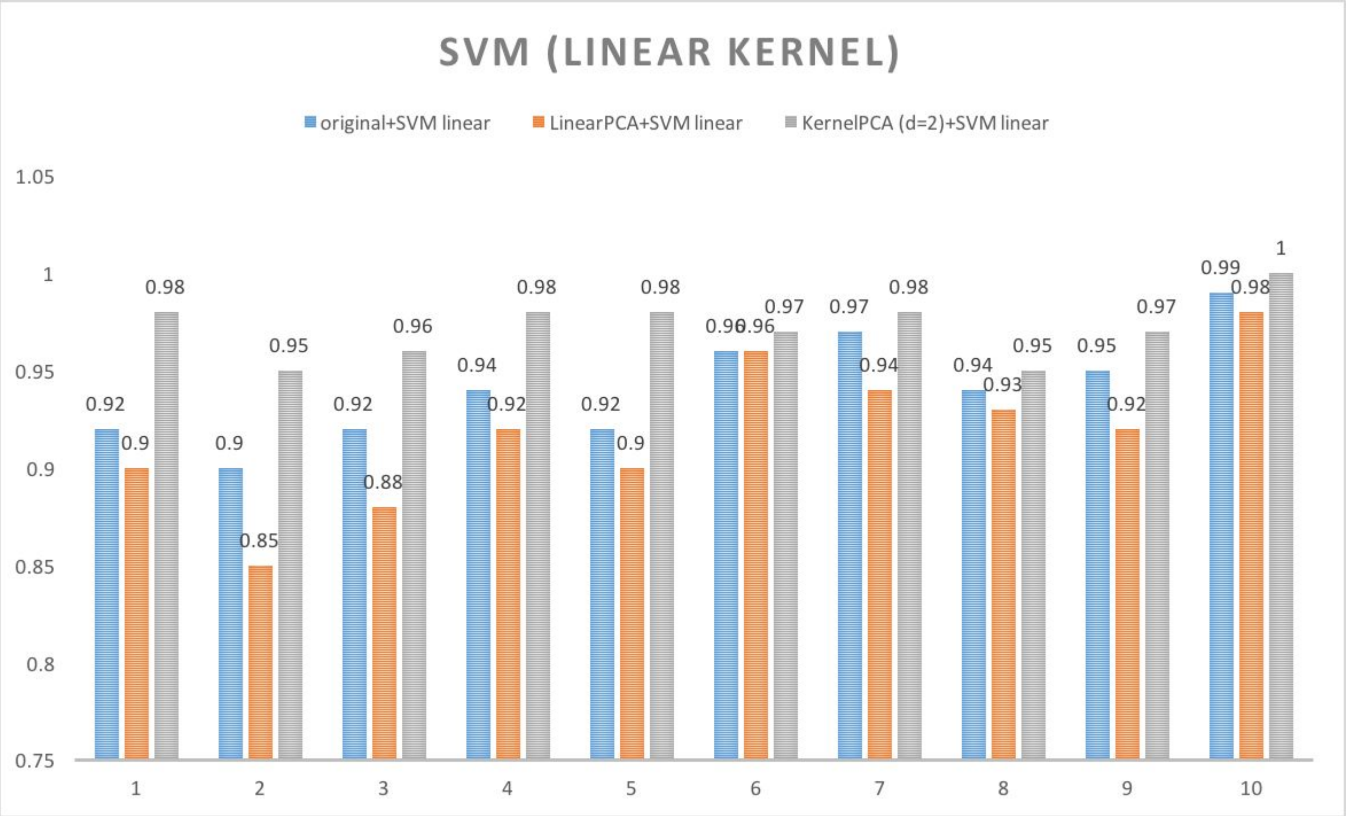


$$\textbf{Precision} = \frac{TP}{TP + FP}$$

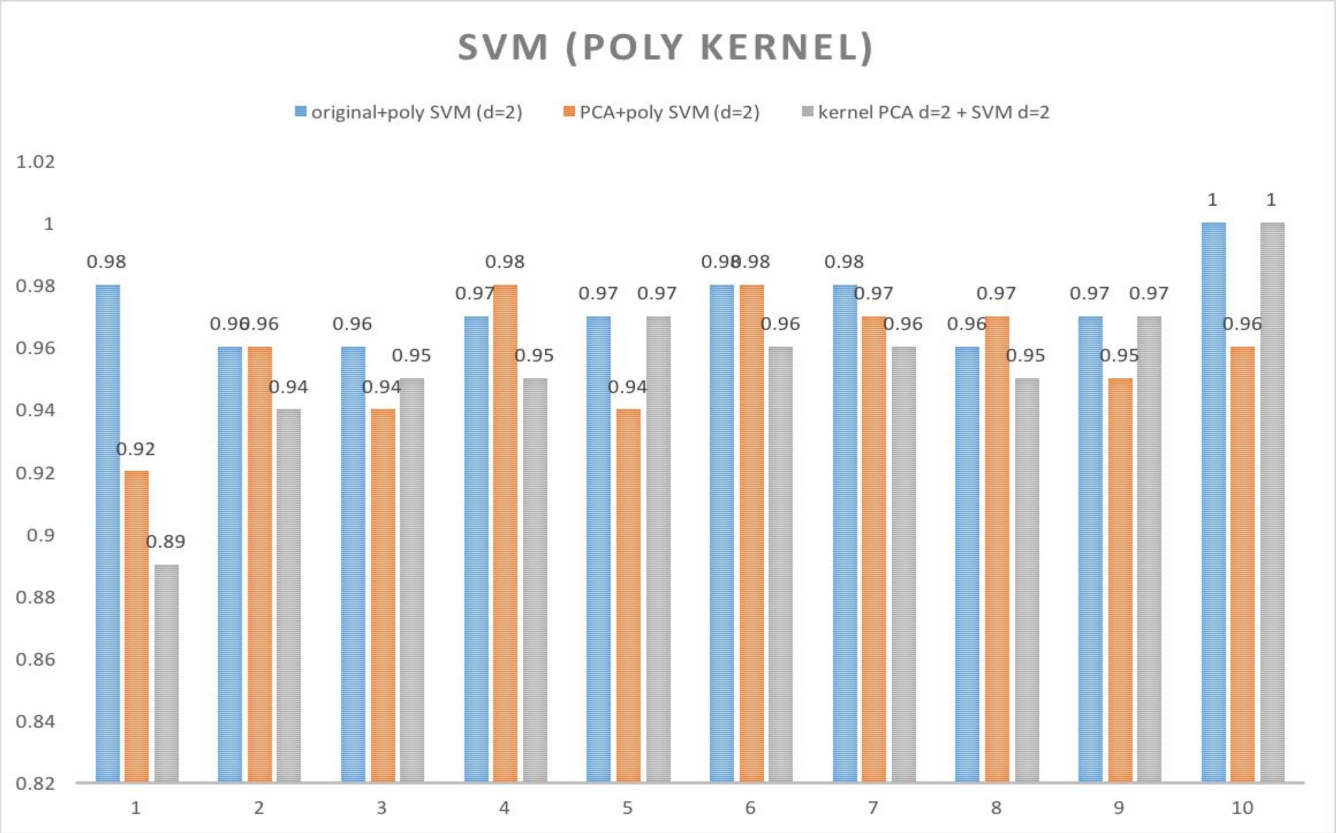
$$\textbf{Recall} = \frac{TP}{TP + FN}$$

$$\textbf{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

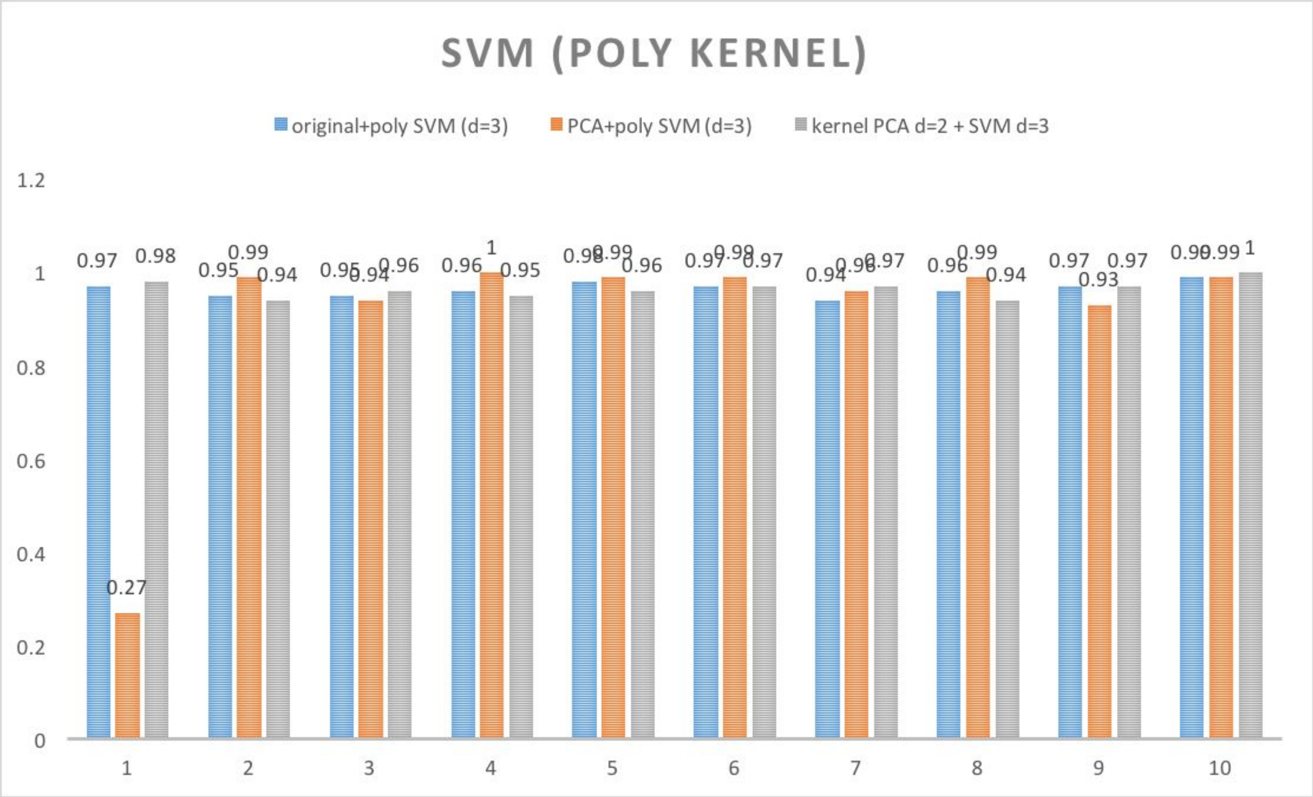
SVM results summary



SVM results summary



SVM results summary



SUMMARY AND COURSE CONNECTION

Course Connection

Principal Component Analysis:

- Able to extract useful features from dataset.
- ‘Kernel method’ : Potentially extract more features than regular PCA.

Clustering:

- More feature not necessarily perform better in visual description of data separation (example : IRIS)

Classification:

- Classifier can predict better if more relevant features are supplied to train.

Summary

ADVANTAGES OF KPCA OVER PCA	DISADVANTAGES OF KPCA OVER PCA
• Able to extract 'M' features (where M is number of obs.) • Able to analyse nonlinear variance. • Classifier has opportunity to train itself better as the extracted feature now depends on number of observations.	• The projection on higher dimensions does not necessarily have a pre-image. • Tough to predict contour lines intuitively. • Clustering (or data separation) does not necessarily work better as extracted feature are abstract in nature.

Summary

- Kernels could be used to find projections on principal components without going though computationally intensive data transformation.
- Kernel method could potentially extract more features as compared to linear PCA.
- Those features capture the maximum variance and hence more representative of the original data.
- Results obtained on linear classifier :
  - Better performance : Higher accuracy
  - Running time : Considerably low as compared to transforming entire data and doing PCA analysis.

REFERENCES

[1] Wang, Quan. "Kernel principal component analysis and its applications in face recognition and active shape models." *arXiv preprint arXiv:1207.3538* (2012).

[2] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem. *"Neural computation 10.5 (1998): 1299-1319.*

[3] Wang, Quan. "Kernel principal component analysis and its applications in face recognition and active shape models." *arXiv preprint arXiv:1207.3538* (2012).

[4] Saegusa, Ryo, Hitoshi Sakano, and Shuji Hashimoto. "A nonlinear principal component analysis of image data." *IEICE TRANSACTIONS on Information and Systems* 88.10 (2005): 2242-2248.