

# Nonlinear Component Analysis as a Kernel Eigenvalue Problem

Jifu Zhao

Huan Yan

Vikram Idiga

Pavan Kumar Nadiminti

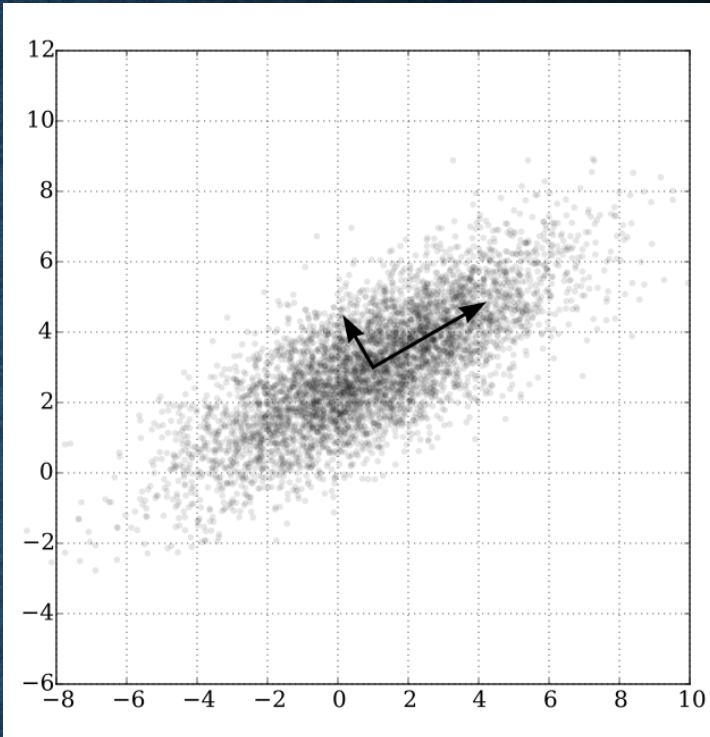
Nov. 16, 2016, Urbana

# Content

- Introduction
  - PCA
  - Kernel PCA
- Algorithms
- Examples
- Summary & Extension

# PCA

- Invented by Karl Pearson, 1901\*.
- Main idea
  - Convert a set of correlated variables linearly uncorrelated variables through orthogonal transformation
- Widely used for dimensionality reduction, feature extraction and data visualization



\* [https://en.wikipedia.org/wiki/Principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis)

# PCA

- Invented by Karl Pearson, 1901\*.
- Main idea
  - Convert a set of correlated variables linearly uncorrelated variables through orthogonal transformation
- Widely used for dimensionality reduction, feature extraction and data visualization

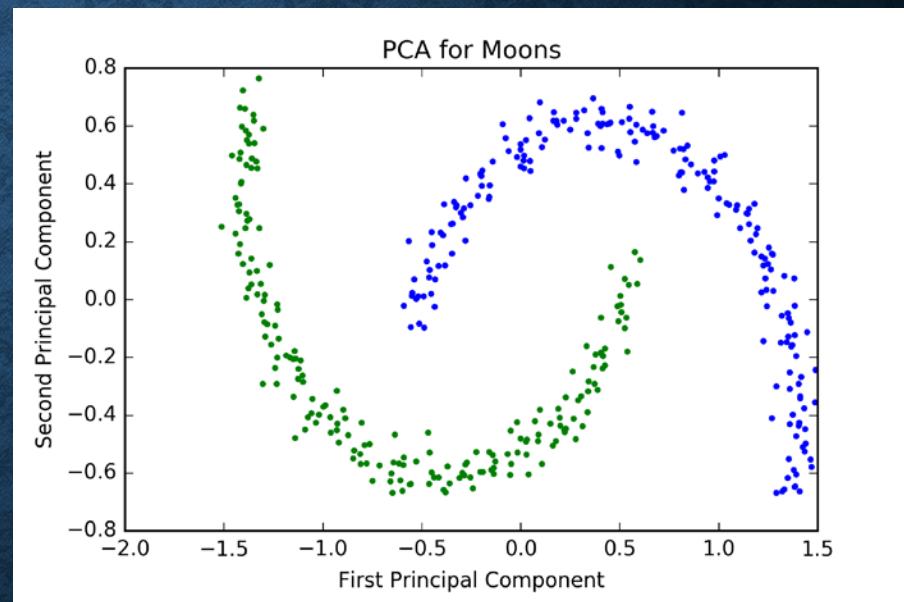
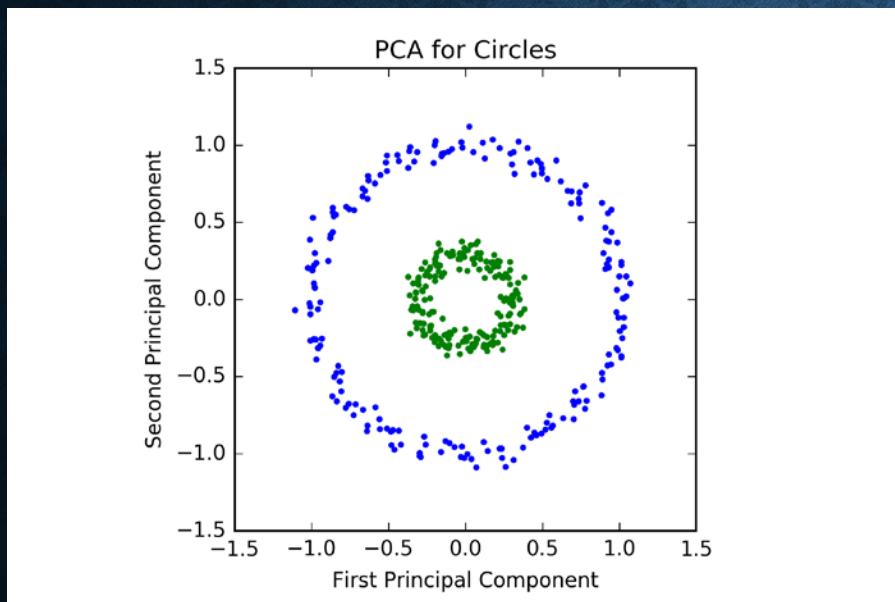
## Algorithm 1 PCA in Feature Spaces

```
1: procedure PCA( $X$ )
2:   given input:  $X_{n \times m} \leftarrow [\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_n]^T$ 
3:   de-mean (or standardize):  $x_{ij} \leftarrow x_{ij} - \bar{x}_j$  or  $x_{ij} \leftarrow \frac{x_{ij} - \bar{x}_j}{s_j}$ 
4:   calculate covariance matrix:  $Cov \leftarrow \frac{1}{n} X^T X$ 
5:   singular value decomposition (SVD):  $[U, S, V] \leftarrow svd(Cov)$ 
6:   choose the first  $k$  eigenvectors:  $E_{m \times k} \leftarrow [\mathbf{u}_1; \mathbf{u}_2; \dots; \mathbf{u}_k]$ 
7:   project the test data  $\mathbf{x}$ :  $\mathbf{p} \leftarrow E^T \mathbf{x}$ 
8: finish
```

\* [https://en.wikipedia.org/wiki/Principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis)

# Why Kernel PCA ?

- PCA works well in a lot of applications
- However, in this simple case, PCA doesn't work at all



- We need some nonlinear transformations

# Kernel PCA

- Put forward by Scholkopf et al., 1998\*
- Main idea
  - Expand the original feature space by non-linear transformations
  - Apply PCA in the transformed feature space
- Main drawback
  - Expanded feature space may have very high dimensions
  - Applying PCA is computationally expensive or even impossible
  - Solution -- Kernels

\* Friedman, J., Hastie, T., & Tibshirani, R. *The elements of statistical learning.*

# Expand the feature space

- Suppose we want to map  $\vec{x}$  into a new space:  $\phi(\vec{x})$
- $\vec{x}$  has  $d$  dimensions and  $\phi(\vec{x})$  has  $m$  dimensions  
$$m > d$$
- We need to calculate:
  - $\vec{x}\vec{x}^T$ : complexity  $O(d^2)$
  - $\phi(\vec{x})\phi(\vec{x})^T$ : complexity  $O(m^2)$
- What if  $m \gg d$  ?
  - Solution: Kernels

# “Kernel Trick”(1)

- **Example:**

- $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$  and  $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$

- $\phi(\vec{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$  and  $\phi(\vec{y}) = \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix}$

- Want to calculate:  $\phi(\vec{x}) \cdot \phi(\vec{y}) = x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2$

- Define  $k(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y})^2$

- $k(\vec{x}, \vec{y}) = x_1^2 y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 = \phi(\vec{x}) \cdot \phi(\vec{y})$

# “Kernel Trick”(1)

- **Example:**

- $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \phi(\vec{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$  and  $\phi(\vec{y}) = \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix}$

- Define  $k(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y})^2$

- $k(\vec{x}, \vec{y}) = x_1^2 y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 = \phi(\vec{x}) \cdot \phi(\vec{y})$

- Important finding:

- Complexity for  $k(\vec{x}, \vec{y})$  is still  $O(d)$  instead of  $O(m)$

- What's next ?

- Find some way to calculate  $k(\vec{x}, \vec{y})$  rather than  $\phi(\vec{x})\phi(\vec{x})^T$

# “Kernel Trick”(2)

- Covariance matrix:

$$C = \frac{1}{n} \sum_{i=1}^m \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T$$

- Find eigenvalues and eigenvectors for:

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

$$\lambda(\phi(\mathbf{x}_k) \cdot \mathbf{v}) = (\phi(\mathbf{x}_k) \cdot C\mathbf{v}) \text{ and } \mathbf{v} = \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i)$$

- For kernel matrix  $K$

$$K_{ij} = (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j))$$

$$n\lambda K\boldsymbol{\alpha} = K^2\boldsymbol{\alpha} \text{ or } n\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}$$

- Projection for any given  $\mathbf{x}$  on  $i$ th principal component

$$p_i(\mathbf{x}) = \sum_{j=1}^n \alpha_{ij} k(\mathbf{x}, \mathbf{x}_j)$$

# Kernel PCA Algorithm

- Conclusion:

- Only need to calculate the kernel matrix  $K$
- find the eigenvalues and eigenvectors for  $K$

# Kernel PCA Algorithm

- Conclusion:

- Only need to calculate the kernel matrix  $K$
- find the eigenvalues and eigenvectors for  $K$

---

## Algorithm 2 Kernel PCA

---

- 1: **procedure** K-PCA( $X$ )
- 2:   given input:  $X_{n \times m} \leftarrow [\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_n]^T$
- 3:   calculate kernel matrix  $K_{n \times n}$ :  $k_{ij} \leftarrow k(\mathbf{x}_i, \mathbf{x}_j)$
- 4:   centralize  $K$ :  $K' \leftarrow K - \mathbb{I}_n K / n - K \mathbb{I}_n / n + \mathbb{I}_n K \mathbb{I}_n / n^2$
- 5:   calculate eigenvector  $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_d$  according to:  $n\lambda\boldsymbol{\alpha} = K'\boldsymbol{\alpha}$
- 6:   normalize eigenvector according to:  $n\lambda_i \boldsymbol{\alpha}_i^T \boldsymbol{\alpha}_i = 1$
- 7:   project the test data  $\mathbf{x}$ :  $p_i(\mathbf{x}) \leftarrow \sum_{j=1}^n \alpha_{ij} k(\mathbf{x}, \mathbf{x}_j)$
- 8: **finish**

---

**Note:**  $\mathbb{I}_n$  stands for  $n \times n$  matrix with all values equal to 1.

# Commonly Used Kernels\*

- Polynomial kernel

$$K(x, y) = (\gamma x^T y + c_0)^d$$

- Radial basis function kernel (Gaussian or RBF kernel)

$$K(x, y) = \exp(-\gamma \|x - y\|^2)$$

- Sigmoid kernel

$$K(x, y) = \tanh(\gamma x^T y + c_0)$$

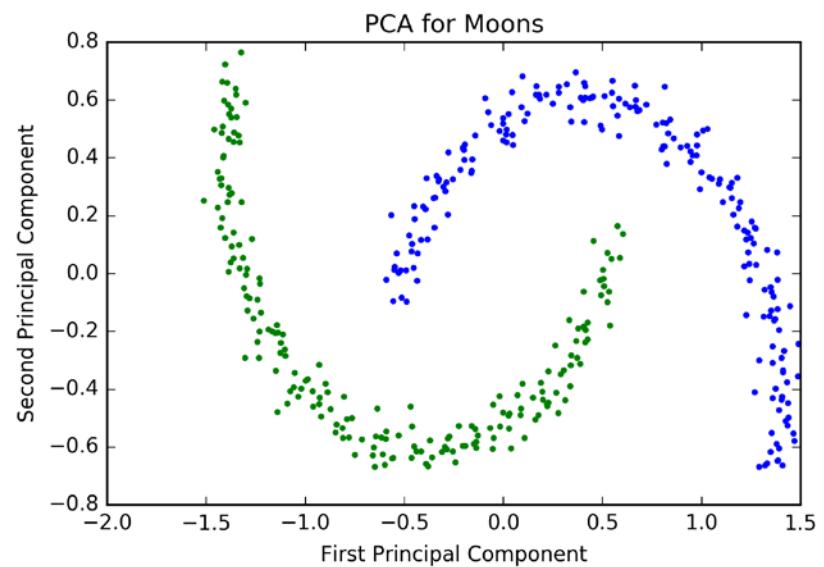
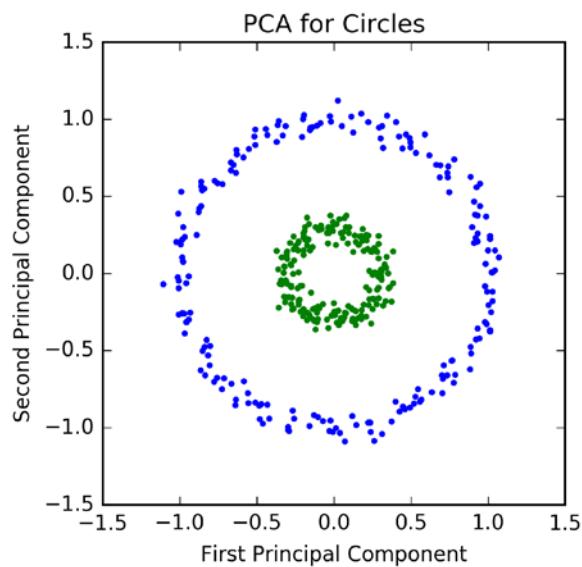
- Laplacian kernel

$$K(x, y) = \exp(-\gamma \|x - y\|_1)$$

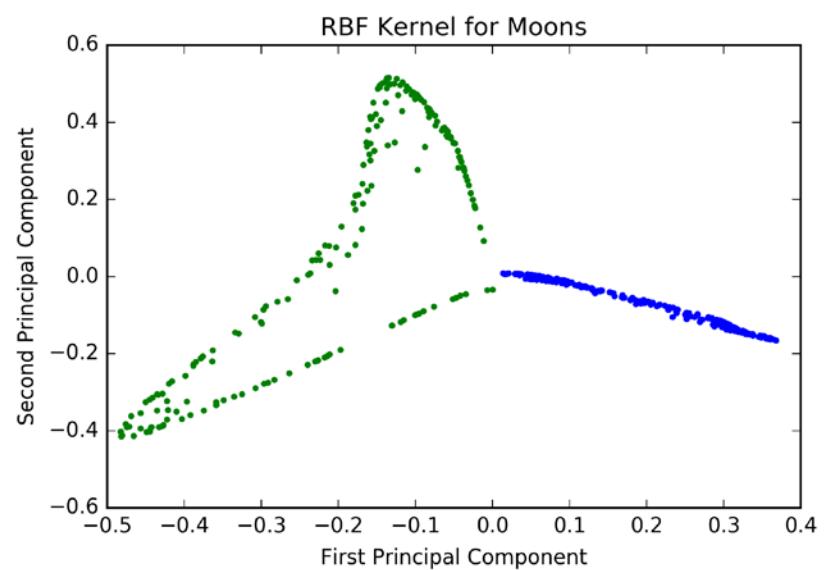
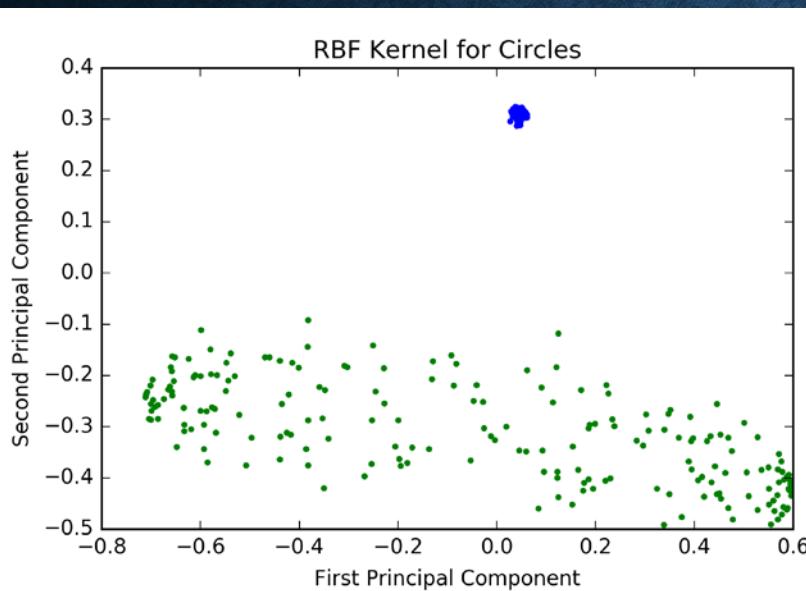
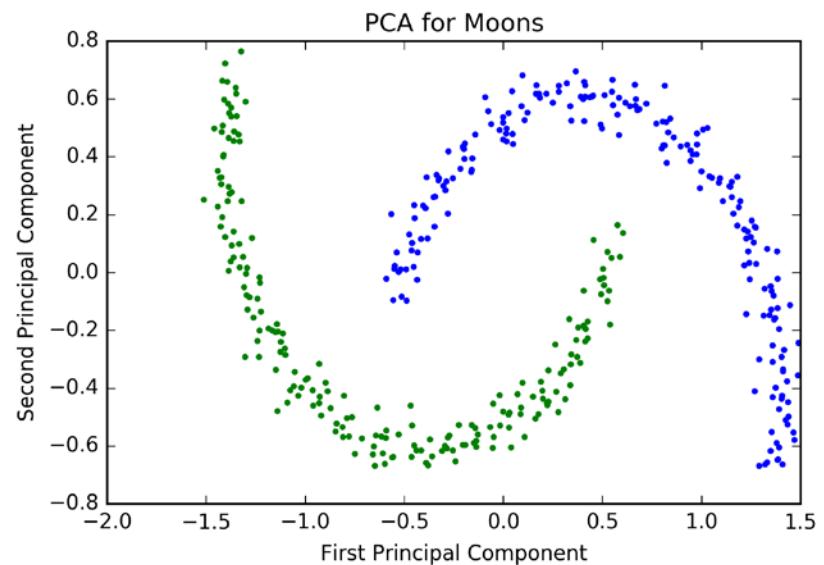
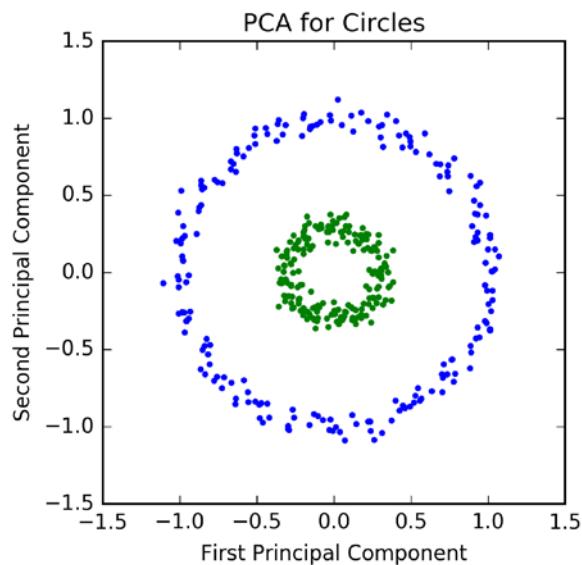
- And so on.

\* <http://scikit-learn.org/stable/modules/metrics.html#metrics>

# Toy Examples



# Toy Examples



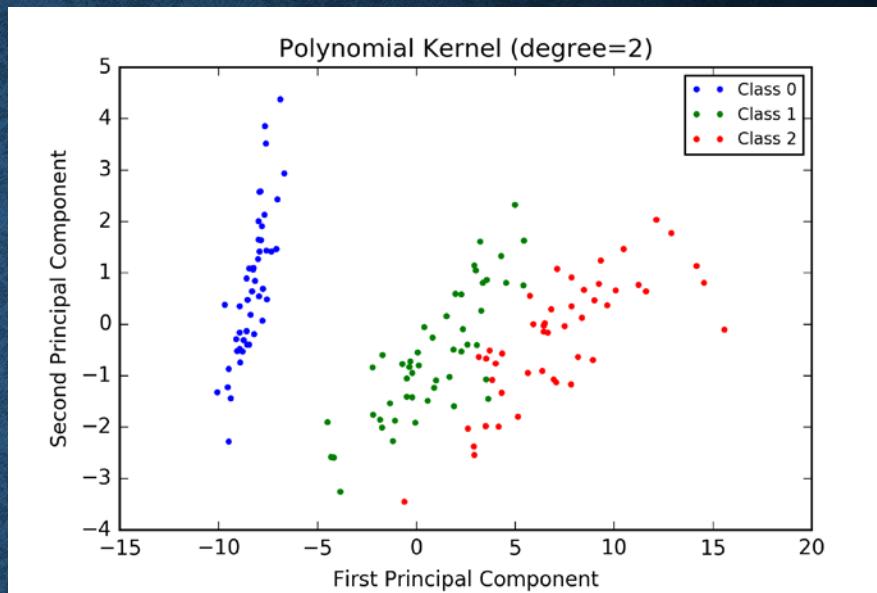
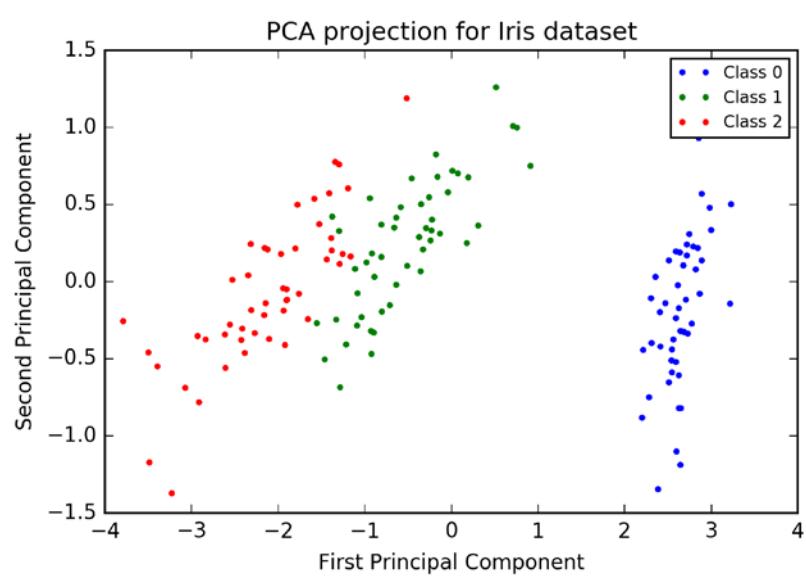
# Iris Dataset – Introduction

- Iris flower dataset is introduced by Ronald Fisher (1936)\*
- It contains 3 different types of irises (Setosa, Versicolor, and Virginica)\*\*.
- The dataset has 150 records and each has 4 features:
  - Sepal Length
  - Sepal Width
  - Petal Length
  - Petal Width.

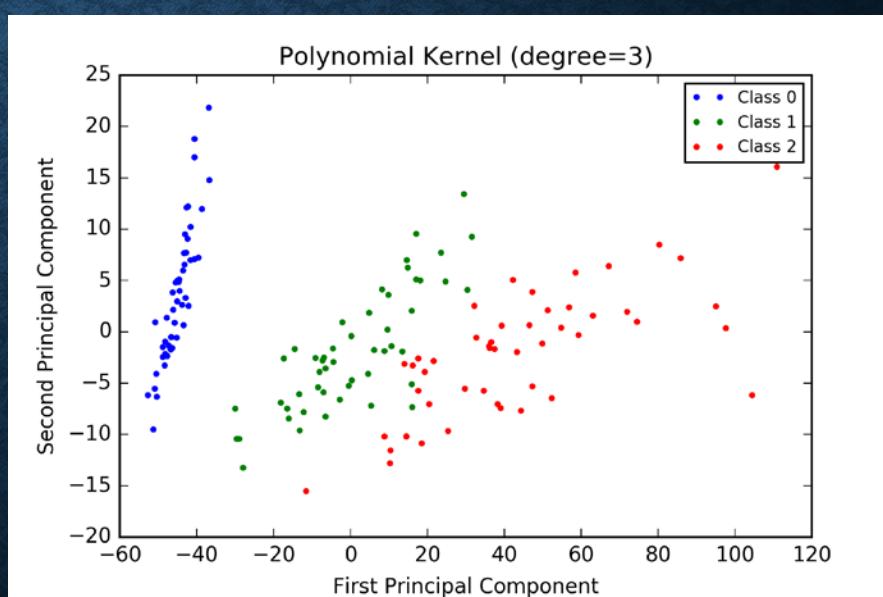
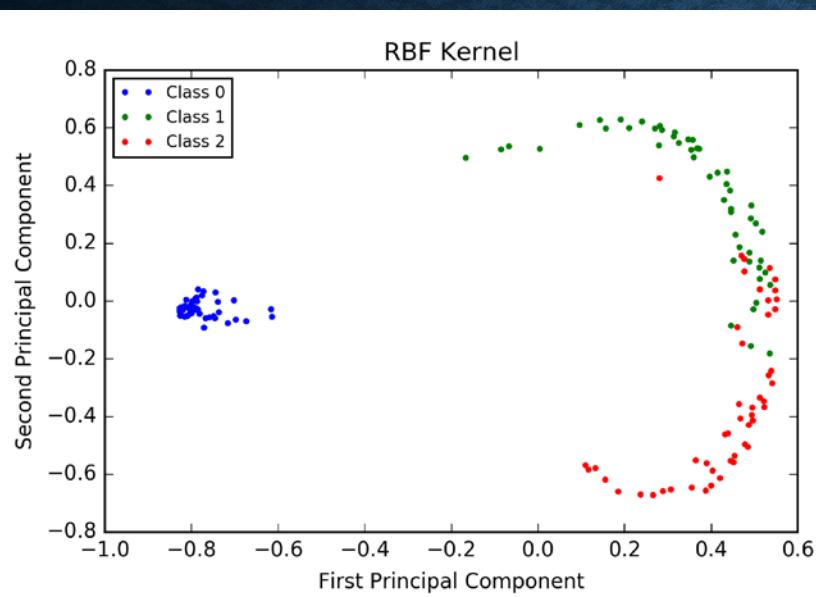
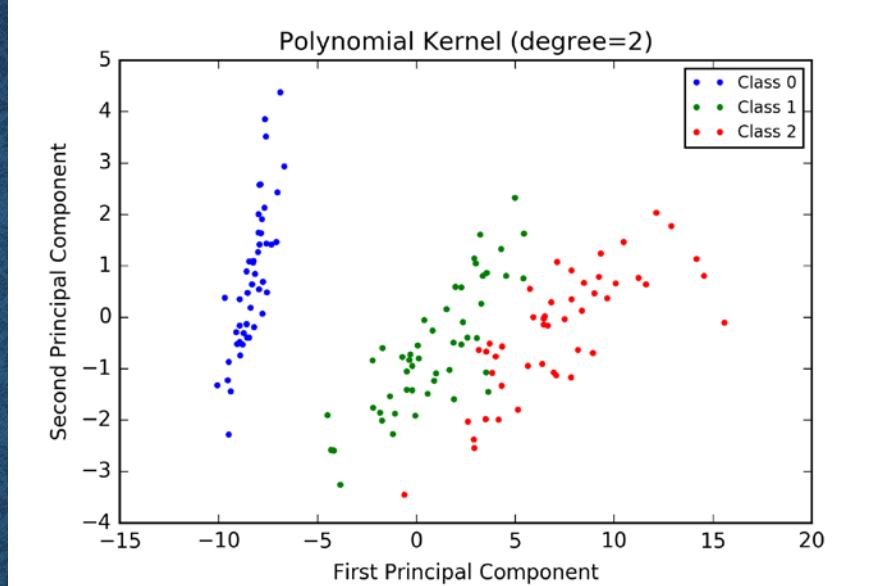
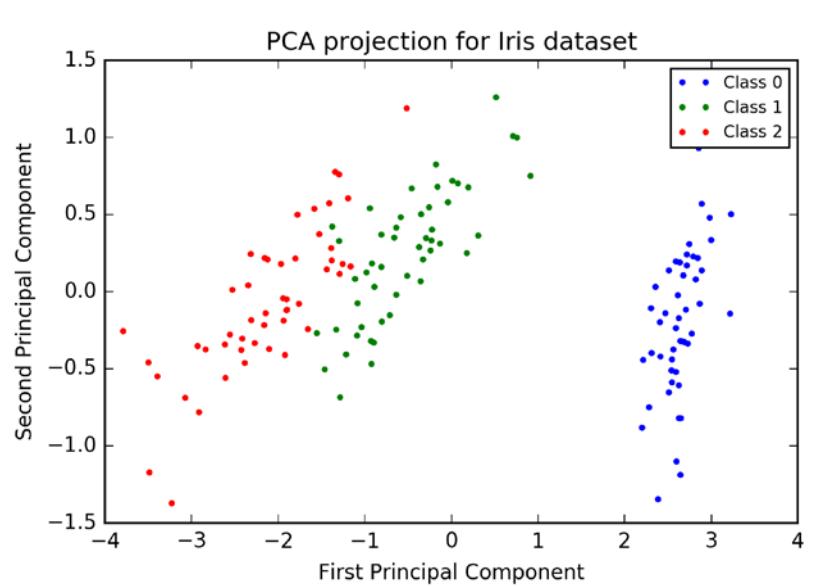
\* [https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set)

\*\* [http://scikit-learn.org/stable/auto\\_examples/datasets/plot\\_iris\\_dataset.html](http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)

# Iris Dataset – PCA vs. K-PCA



# Iris Dataset – PCA vs. K-PCA



# Digits Dataset – Introduction

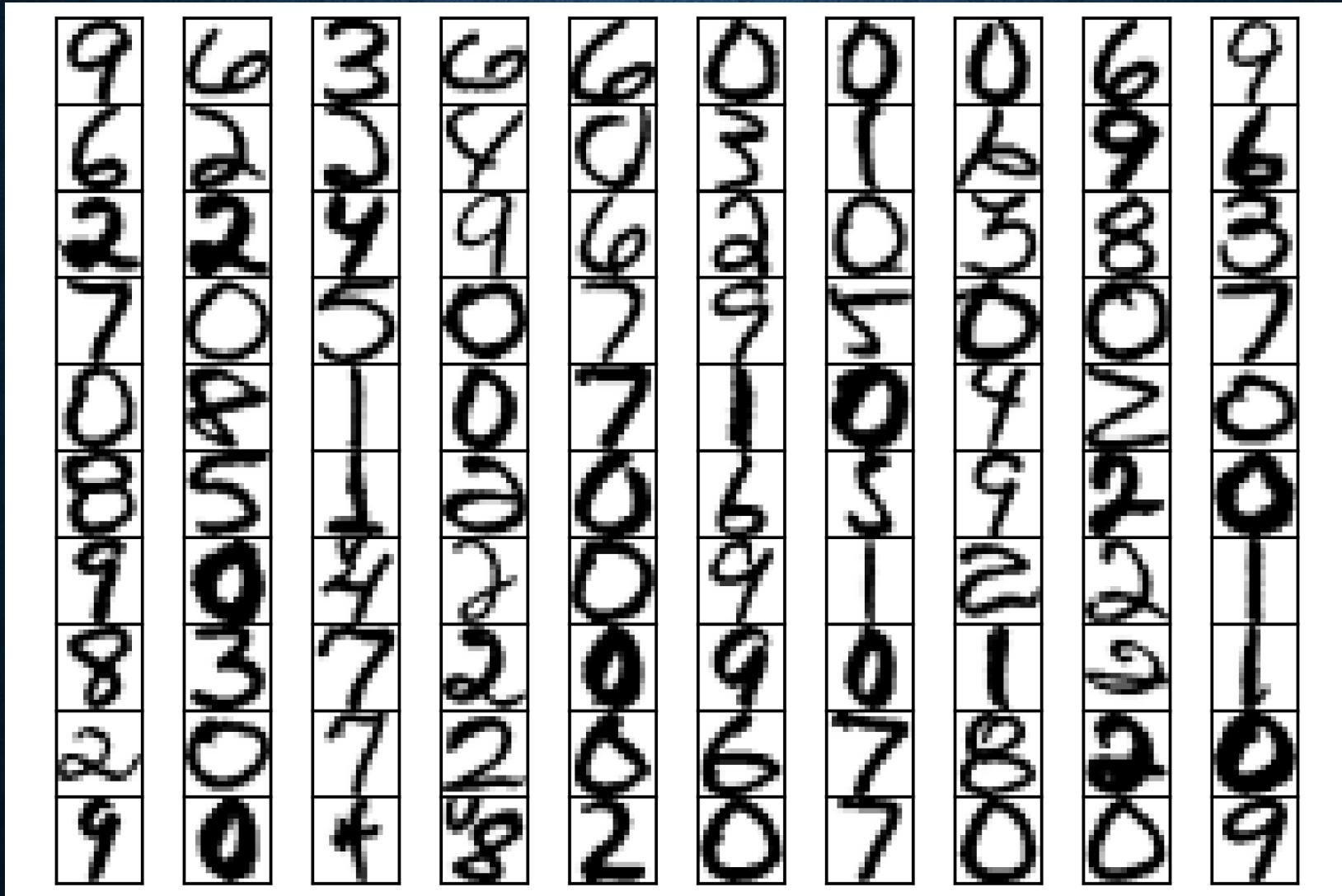
- Normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service\*.
- All the images have been processed into  $16 \times 16$  grayscale images (256 features).
- The dataset has 10 classes (0-9). There are 7291 training observations and 2007 test observations\*\*.

Class	0	1	2	3	4	5	6	7	8	9	Total
Train	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007

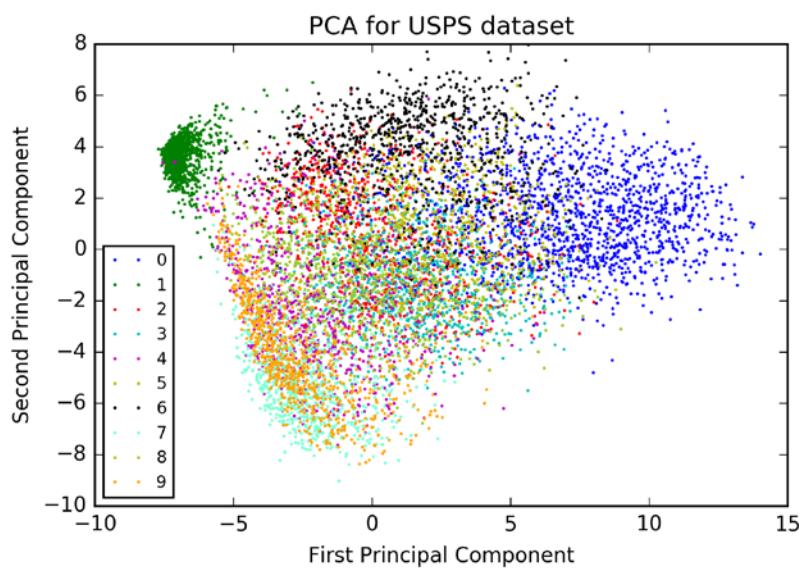
\* <http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.info.txt>

\*\* <https://www.otexts.org/1577>

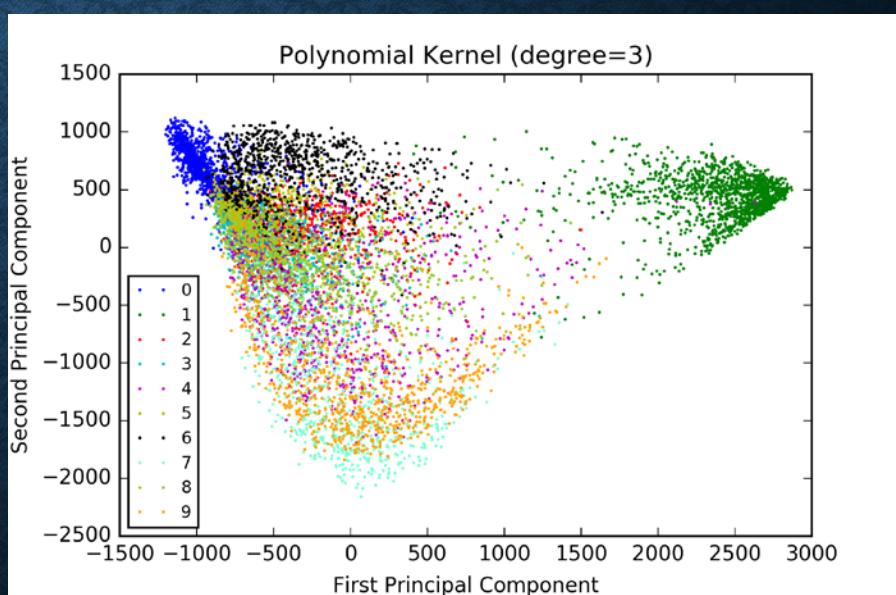
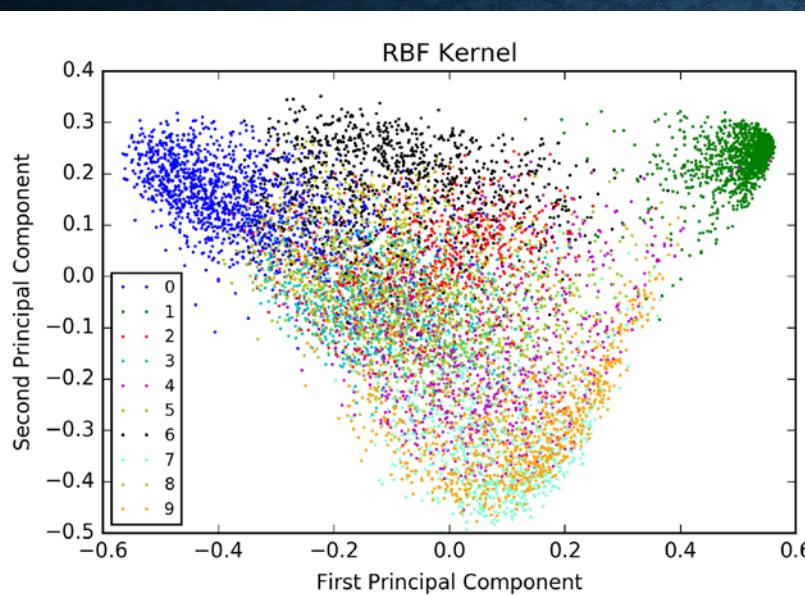
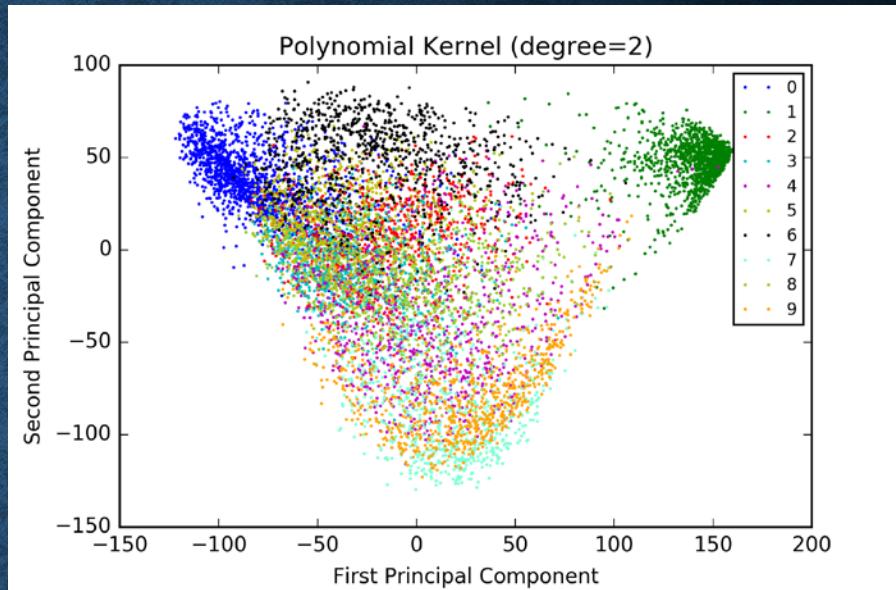
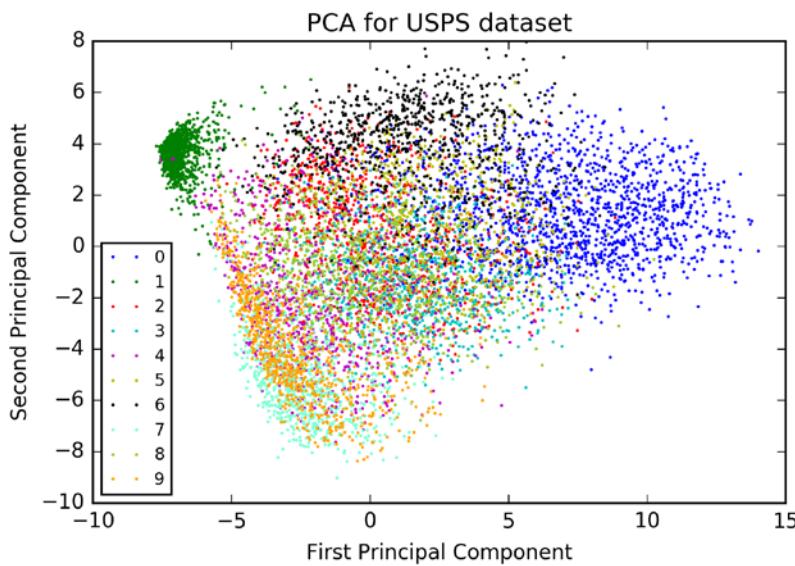
# Digits Dataset – Introduction



# Digits Dataset – PCA vs. K-PCA

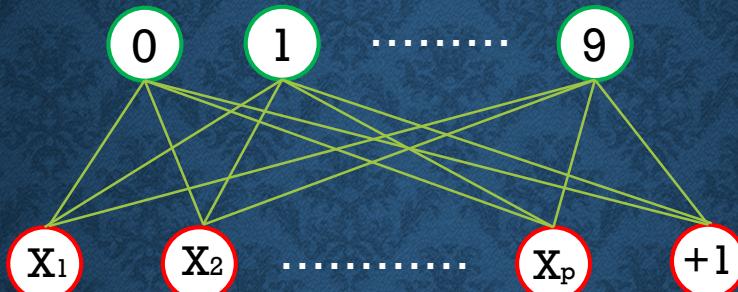


# Digits Dataset – PCA vs. K-PCA



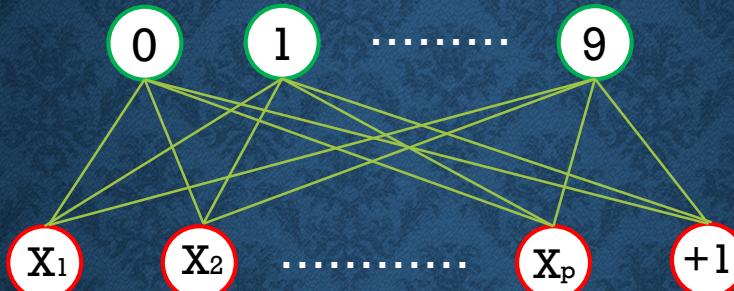
# Digits Dataset – Classification

- Train a 10-way multi-class neural network for classification



# Digits Dataset – Classification

- Train a 10-way multi-class neural network for classification



- Training accuracy and testing accuracy

Inputs	Raw image	PCA	Polynomial Kernel (degree=2)	Polynomial Kernel (degree=3)	RBF Kernel
Feature Number	256	128	512	1024	1024
Training Accuracy	95.52%	94.94%	99.99%	98.51%	93.58%
Test Accuracy	90.98%	90.23%	94.42%	93.92%	88.14%

# Summary

- Computation complexity comparison
  - PCA:  $\sim O(nm^2 + m^3)$
  - Kernel PCA:  $\sim O(n^2m + n^3)$
- Kernel PCA takes advantage of kernels to avoid expensive computations
- Kernels PCA can work better than PCA for some problems
- Need to note:
  - Kernel matrix  $K$  has dimension of  $n \times n$ . It will be expensive to find eigenvectors when  $n$  is large ( $\sim O(n^3)$ )
  - Using kernels, the parameters like  $d, \gamma, c_0$  need to be determined by users

# Extension

- Dimensionality reduction is more generally used
- There are a lot of methods for dimensionality reduction
  - Independent Component Analysis (ICA)
  - Non-negative Matrix Factorization (NMF)
  - Isometric Feature Mapping (ISOMAP)
  - Locality Sensitive Hashing (LSH)
  - Latent Semantic Analysis (LSA)
  - Restricted Boltzmann Machine (RBM)
  - Auto-encoder
  - And so on

# Questions?

- Language:
  - Python + TensorFlow
- All source codes are available in GitHub
  - [https://github.com/JifuZhao/UIUC\\_Courses/tree/master/UIUC\\_IE529/project/code](https://github.com/JifuZhao/UIUC_Courses/tree/master/UIUC_IE529/project/code)
- Thank you.

# References

- [https://en.wikipedia.org/wiki/Principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis)
- <http://scikit-learn.org/stable/modules/metrics.html#metrics>
- [https://en.wikipedia.org/wiki/Iris flower data set](https://en.wikipedia.org/wiki/Iris_flower_data_set)
- [http://scikit-learn.org/stable/auto\\_examples/datasets/plot\\_iris\\_dataset.html](http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)
- <https://www.otexts.org/1577>
- <http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.info.txt>
- Schölkopf, B., Smola, A. & Müller, K.R., 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*.
- Friedman, J., Hastie, T., & Tibshirani, R. *The elements of statistical learning*.
- Bishop, C. M. *Pattern Recognition and Machine Learning*.