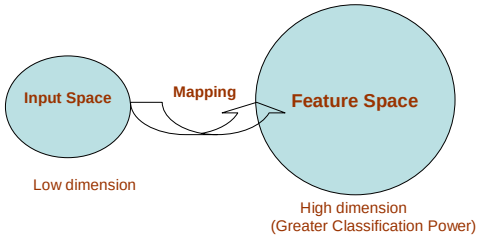
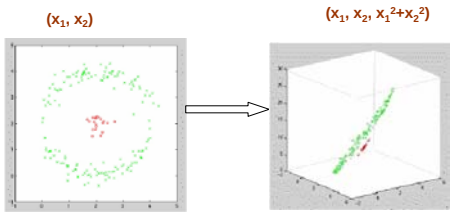
Improved Kernel PCA for Feature Extraction

Xianhua Jiang
Yuichi Motai
09/06/05

VC(Vapnik-Chervonenkis) Theory



Ex.: Two Class Problem with Data in R^2



Kernel Trick

- Most classification method can be represented as inner product, such as PCA, SVM.
- Kernel makes inner product in high-dimension done in low dimension

$$k(x, y) = <\Phi(x), \Phi(y)>$$

Kernel

Polynomial Kernel

$$k(x, y) = [< x, y > +1]^d$$

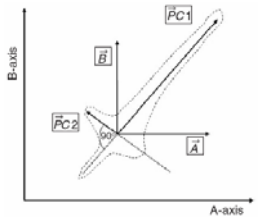
Radial basis function Kernel

$$k(x, y) = \exp\{-r \| x - y \|\}$$

Linear PCA

- Find orthonormal axes to maximally decorrelate the data
- Assumption:
  - ❑ Sources are Gaussian
  - ❑ Sources are independent and stationary.

Ex.: PCA



Standard PCA Algorithm

- Centered Observations: column vectors, $x_i \in R^l$ $i = 1, \ldots, n$
- PCA finds the principal axes by diagonalizing the Scatter matrix

$$S = \sum_{i=1}^{n} x_i x_i^T$$

- Note that S is positive definite, and thus can be diagonalized with nonnegative eigenvalues.

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

Using PCA

- Find eigenvectors, and arrange in order of decreasing eigenvalue.
- Project test points onto eigenvectors
- use those coefficients to do something useful (classification,image reconstruction, etc).

PCA in Mapped Feature Space

$$S = \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T$$

$$\lambda e_j = Se_j = \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T e_j = \sum_{i=1}^{n} (< e_j, \Phi(x_i) > \Phi(x_i))$$

$$e_j = \sum_{i=1}^{n} a_{ji} \Phi(x_i)$$

Kernel PCA

Calculate Coefficient α_i:

$$\lambda a_j = K a_j$$

Where: $k_{ij} = <\Phi(x_i), \Phi(x_j)> = <x_i, x_j>^d$

Represent the principal vector:

$$e_j = \sum_{i=1}^{n} a_{ji} \Phi(x_i)$$

Projection to the principal vector:

$$< e_j, \Phi(x) > = \sum_{i=1}^{n} a_{ji} < \Phi(x_i), \Phi(x) > = \sum_{i=1}^{n} a_{ji} < x, x_i >^d$$

Comment on Kernel PCA

- Calculate the dot product at feature space without calculating the mapping Φ(x).
- Need to solve n*n eigenvalue problem, which is time-consuming.
- Each eigenvector is the combination of all training data, which makes it unacceptable for on-line learning

Improve Kernel PCA

- Select a subset of input data, then apply kernel PCA.
- Selected subset of data are important in feature space according to the Kernel we chosen, which is similar as clustering.
- Random sampling is must faster, and have equal performance in some cases. (It's very practical, but too simple to write paper!)

Kernel Feature Analysis

Extracts the eigenvectors one by one according to the decreasing order of eigenvalues.

$V_{LP} = \left\{ w \middle| w = \sum_{j=1}^{n} a_j \Phi_j \; with \sum_{i=1}^{n} |a_i| \le 1 \right\}$ has a vertex solution.

KFA Algorithm

1. Calculate n*n Gram matrix K
2. Orthogonalize the possible directions to any previous principal vector
3. Compute the dot products between the possible direction and the training data
4. Compute the variance of each possible directions
5. Choose the direction with the maximum variance as the principal vector
6. Normalize this principal vector

Commend On KFA

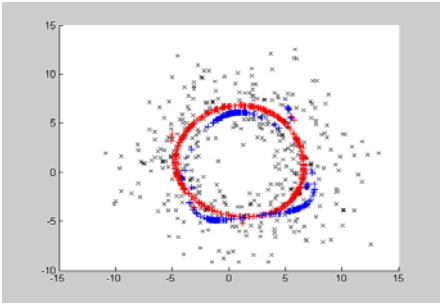
- KFA extracted *P* eigenvectors, where in most cases *P* much less than n.
- Computational complexity of computing *P* features is $O(P \times n^2)$, while the standard kernel PCA costs $O(n^3)$.
- Orthogonalize, normalize, calculate the projection variance, sort, then unitize the eigenvectors, it didn't save much time as expected.
- The eigenvectors of KFA are approximate solutions. Its performance is not comparable with Kernel PCA.

Improved KFA

- After select one principle vector, discard amount of data that are not potential new principle vectors.
- Increase the efficiency, but not helpful to improve its performance.

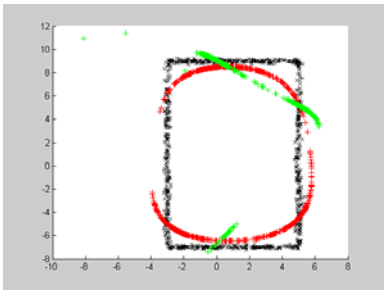
Experiment Results

400 points, 2 principle vectors, RBF Kernel



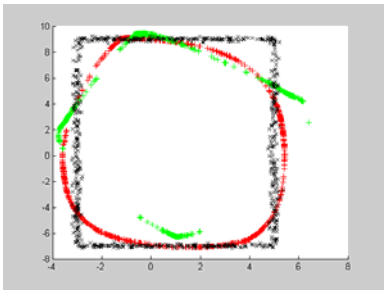
R: KPCA
B: KFA

800 points, 2 Eigenvectors, RBF Kernel



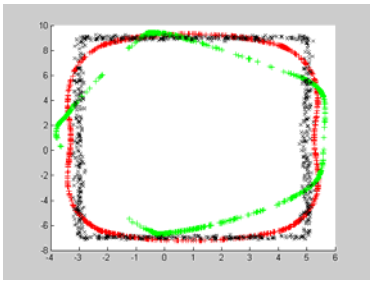
R: KPCA
G: KFA

800 points, 3 Eigenvectors, RBF Kernel



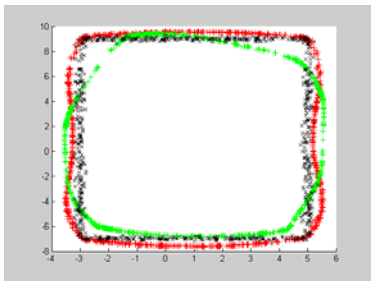
R: KPCA
G: KFA

800 points, 4 Eigenvectors, RBF Kernel



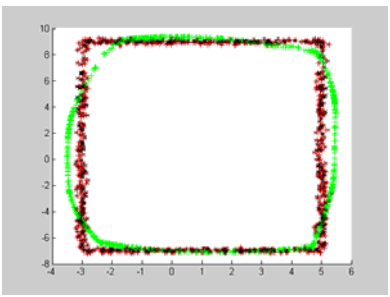
R: KPCA
G: KFA

800 points, 10 Eigenvectors, RBF Kernel



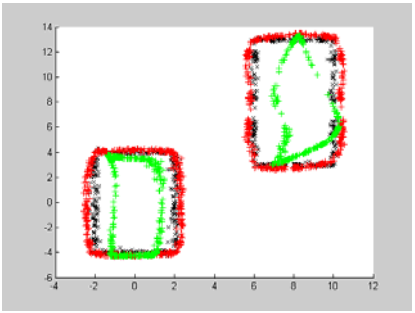
R: KPCA
G: KFA

800 points, 20 Eigenvectors, RBF Kernel



R: KPCA
G: KFA

Each 400 points, 10 Eigenvectors, RBF Kernel



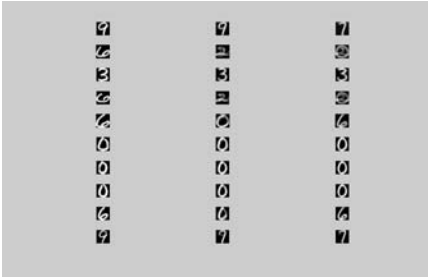
R: KPCA
G: KFA

2000 points (within one rectangle), 10 Eigenvectors, RBF Kernel

	Recons. Error In High Dim.	Cpu-time(s)	Data Related
KPCA	1.2572e-004	375.2496	All
KFA	5.5607e-004	331.7971	[1582 1146 331 1051 1176 101 325 1144 152 1684]
Improved KFA	6.2320e-004	480.5810	[1582 1146 331 1051 1176 101 325 1144 152 1470]

USPS Zip Code. 1000 training data. 10 test data. 10 Eigenvectors, RBF kernel

Original Image Pre-image of KPCA Pre-image of KFA



Simple Classification

1. To each digit, use 300 samples to train, store 10 eigenvectors. Then project the test data to each eigenspace, choose the eigenspace with minimum reconstruction error as the class it belongs to.
- Use 100 data to test, Accuracy of KPCA is 91%, accuracy of KFA is 84 %

300 training data for each digit, 100 test data

Test data+ random noise

Accuracy (%)	Number of Principle Vectors Extracted in Feature Space					
	5	10	20	50	100	150
Kfa	68	71	75	78	78	78
kpca	81	88	87	88	91	91

Training data+ random noise, Test data+ random noise

Accuracy (%)	Number of Principle Vectors Extracted in Feature Space					
	5	10	20	50	100	150
Kfa	81	81	80	79	82	85
kpca	88	89	91	91	87	90

300 training data for each digit, 100 test data

Training data + random noise

Accuracy (%)	Number of Principle Vectors Extracted in Feature Space						
	5	10	20	50	100	150	200
Kfa	40	44	46	45	50	51	50
kpca	54	53	58	62	65	57	63

No noise both to training data and test data

Accuracy (%)	Number of Principle Vectors Extracted in Feature Space						
	5	10	20	50	100	150	200
Kfa	82	84	86	91	90	90	90
kpca	90	91	91	93	93	93	93

Next Step

- Give up the instance selection for kernel PCA
- Try to improve the performance of KFA, such as making each principle vectors contain more data information.
- From the theoretical view, analyze the difference of the eigenvectors by Kernel PCA and by Kernel Feature Analysis.
- From the theoretical view, understand the property of kernel, the property of the high dimension.
- From the theoretical view, how to choose kernel, whether we can analyze data can be linearly separated in the high dimension without experiment?

Acknowledge

Prof.Motai
Prof.Zhu
Prof.Snapp

Thank you!