Group Project Summary Report for: Nonlinear Component Analysis as a Kernel Eigenvalue Problem

Huan Yan, Jifu Zhao, Pavan Kumar Nadiminti, and Vikram Idiga

November 7, 2016

1 Introduction

In this paper, the authors proposed a new approach to a traditional problem of identifying major modes/components of a given dataset. Traditionally, Principal Component Analysis (PCA) is used to detect major structure within data. Essentially it is an eigenvalue problem in which we transform the coordinate axes in such a way that the variances of the data set we have, lie mainly along those axes. These directions are called as principal components. PCA assume that there is linear denpendency among data, which is not the case for some dataset. And in some tasks, such as image analysis, it would be necessary to extract features that are non-linearly related to the input variables from a higher dimensional space. To do so we can follow the tradditional PCA approch after mapping the input variables to the desired feature space. However, if the dimension of the desired feature space is very high, the computation of mapping and covariance matrics can be computationally intractable. To avoid this problem, the author provide an approch, which is different from the traditional covariance matrix method and take advantage of kernel function to perform non-linear PCA in high dimensional feature space. The kernel functions are essentially the dot products in the feature space transformed by the non-linearity relating the input space to the desired feature space. Using kernel functions we can directly find out the dot product in very high dimensional space without doing any mapping, which in some cases can be very computationally expensive. Meanwhile, this kernel PCA approach avoid solve eigenvalue problem on covariance matrix. Instead, it find eigenvalues and eigenvectors of kernel matrix, which can be much easier if the dimension of feature space is very high compared to the size of dataset.

2 Feature Space PCA

The entire analysis is done on centered data. Here we perform PCA analysis and explore ways to represent the PCA equations in terms of dot products. Basic PCA involves computing the covariance matrix:

$$C = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{x}_j^T \tag{1}$$

And solving the eigenvalue problem:

$$\lambda \boldsymbol{v} = C\boldsymbol{v} \tag{2}$$

Following the above procedure, the detailed PCA algorithms is shown below:

Algorithm 1 PCA in Feature Spaces

```
1: procedure PCA(X)
2:     given input: $X_{n\times m} \leftarrow \begin{bmatrix}\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_n\end{bmatrix}^T$
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: $Cov \leftarrow \frac{1}{n}X^TX$
5:     singular value decomposition (SVD): $[U, S, V] \leftarrow svd(Cov)$
6:     choose the first k eigenvectors: $E_{m\times k} \leftarrow \begin{bmatrix}\mathbf{u}_1; \mathbf{u}_2; \cdots; \mathbf{u}_k\end{bmatrix}$
7:     project the test data $\mathbf{x}$ : $\mathbf{p} \leftarrow E^T\mathbf{x}$
8: finish
```

3 Kernel PCA

The linear PCA presented above cannot deal with the non-linearity in dataset. In this case we might want to map the data to a higher dimenstional feature space by: $\phi : \mathbf{R}^m \to F$. We then perform the PCA in the new space.

Following the traditional PCA approch, we find the covariance matrix and the corresponding eigenvalue equation in the feature space can be expressed as:

$$\bar{C} = \frac{1}{n}\sum_{j=1}^{n}\phi(\boldsymbol{x}_j)\phi(\boldsymbol{x}_j)^T \tag{3}$$

$$\lambda\boldsymbol{V} = \bar{C}\boldsymbol{V} \tag{4}$$

However, as we state, this eigenvalue problem can be computationally expensive to solve if the dimension of F is very high. So we modify Eq.4: The eigenvalue problem $\lambda\boldsymbol{V} = \bar{C}\boldsymbol{V}$ can also be expressed in terms of a dot product as follows:

$$\lambda(\phi(\boldsymbol{x}_k)\cdot\boldsymbol{V}) = (\phi(\boldsymbol{x}_k)\cdot\bar{C}\boldsymbol{V}) \tag{5}$$

for all k = 1, ..., n

Another important step is that we expressed the eigenvector $\boldsymbol{V}$ in terms of linear combination of feature vectors:

$$\boldsymbol{V} = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i) \tag{6}$$

Now our eigenvalue problem can be expressed as dot products of transformed feature vectors, here is where kernel functions come into play.

$$\lambda\sum_{i=1}^{n}\alpha_i(\phi(\boldsymbol{x}_k)\cdot\phi(\boldsymbol{x}_i)) = \frac{1}{n}\sum_{i=1}^{n}\alpha_i(\phi(\boldsymbol{x}_k)\cdot\lambda\sum_{i=1}^{n}\phi(\boldsymbol{x}_i))(\phi(\boldsymbol{x}_j)\cdot\phi(\boldsymbol{x}_i)) \tag{7}$$

Mercers Theorem states that if $k(\boldsymbol{x},\boldsymbol{y})$ is a continuous kernel of a positive integral operator, there exists a mapping where k acts as a dot product. Dot product in new space is represented as a kernel:

$$k(\boldsymbol{x},\boldsymbol{y}) = (\phi(\boldsymbol{x})\cdot\phi(\boldsymbol{y})) \tag{8}$$

Then the eigenvalue problem can be further simplified as:

$$n\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha} \tag{9}$$

Where the kernel matrix K is defined as:

$$K_{ij} := (\phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x}_j)) \tag{10}$$

Then the eigenvectors and eigenvalues can be found by solving this equation.

As we can see, in kernel PCA, a non-trivial arbitrary function $\phi()$ is chosen, but is never calculated explicitly, allowing the possibility to use a very high dimensional $\phi()$. Hence, we avoid computing the eigenvectors and eigenvalues of the covariance matrix in the high dimensional feature space. Instead, we find the kernel matrix and compute its eigenvectors and eigenvalues, which in certain case can be much easier.

The projection of feature vectors onto principal components can be calculated in the following way:

$$(\boldsymbol{V}^i \cdot \phi(\boldsymbol{x})) = \sum_{j=1}^{n} \alpha_{ij} k(\mathbf{x}, \mathbf{x}_j) \tag{11}$$

Where $\boldsymbol{\alpha}_i$ is the eigenvector of K, and λ_1, λ_2, ..., λ_m, are the eigenvalues of K.

The algorithm for Kernel PCA is summarized below:

Algorithm 2 Kernel PCA

```
1: procedure K-PCA(X)
2:     given input: $X_{n\times m} \leftarrow \begin{bmatrix} \mathbf{x}_1; \mathbf{x}_2; \cdots ; \mathbf{x}_n \end{bmatrix}^T$
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 $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_d$ 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.

4 Properties of Kernel PCA

The salient feature of the Kernel PCA is that when we are extracting the non-linear features from the data in a very high dimensional space, we can avoid mapping step and eigenvalue decomposition of covariance matrix, which can be very difficult or even impossible. The usual properties of PCA continue to hold here, such as: The first q principal components (the projections of eigenvectors) carry more variance than any other q orthogonal directions.

The mean squared error approximation is representing the observations by first q principal components is minimal. The principal components explain most of the variation in the data.

5 Computational Complexity of the Kernel PCA

Kernel PCA can be computationally intensive when compared to its linear counterpart. Suppose we have n samples and each has m dimensions. The computation complexity for normal PCA is $\boldsymbol{O}(nm^2 + m^3)$, and for kernel PCA it's $\boldsymbol{O}(n^2m + n^3)$. As we can see the kernel PCA can handle high dimensional feature space, but when the dataset is big, the computation can be expensive. On the other hand the linear PCA, if performed in higher dimensional space, can be more complex. However, if we consider the fact that Kernel PCA can extract non-linear features from higher dimensional space with comparable computation effort, it is still a superior method in many cases. And also we need to point out that, in feature extraction for further classification, the more effort spent in kernel PCA can be compensated by much simpler classification process since in the higher dimensional space the data would be linearly seperable.

6 Experiments by the Authors

To explore the effectiveness of the Kernel PCA the authors explore two examples cases.

6.1 Toy example

The authors generate an artificial 2-d data set using kernel of varying degrees. Linear PCA produces only two modes for only two non-zero eigenvalues as the data is in 2 dimensions. In contrast the nonlinear PCA allows more components to be discerned. Also the first component from non-linear PCA fits the data better than the linear one. This method also better explained how different principle components extracted using Gaussian kernels separate the clusters in the input data.

6.2 Character Recognition Example

A data set of hand written data from USPS is used to perform K-PCA and then the components utility is assessed to classify the images using character recognition classifier. The performance of nonlinear PCA is found to be better than that of linear PCA. The main reason for this is that there are many higher order features in an image than there are pixels in it. Nonlinear PCA addresses this issue better.

7 Advantages of Kernel PCA

- Kernel PCA can extract non-linear feature from data with comparable computational effort. These non-linear features are important in some tasks.
- Kernel PCA is simpler compared to other non-linear feature extraction methods, which might require nonlinear optimization. It only requires solving eigenvalue problem of kernel matrix.
- Kernel PCA can cover a wide range of non-linearity using different kernels.

8 Experiments in This Project

In our project, we will carry out two experiments on two different datasets.

8.1 Iris Flower Dataset

The first dataset we used is the iris flower dataset, which contains four features of three different iris flower species: Iris setosa, Iris virginica and Iris versicolor. The One of the species (Iris setosa) is linearly separable with others, while the other two species are not linearly separable. Kernel PCA can handle linearly non-separable data in higher dimensional feature space, and can be easily applied on this very small dataset with only 150 samples.

8.2 USPS Dataset

For the second experiment we will use the same dataset used in this paper. This dataset includes more than 9000 of 16×16 grey-scale images, which came from scanning of handwritten digits on the USPS envelops. We will first use kernel PCA to extract feature from images and then apply a neural network with one hidden layer to perform classification. Then the accuracy will be compared with the same neural net trained with features extracted by linear PCA.