Group Project Summary Report on Paper: Non-Linear Component Analysis as a Kernel Eigen Value Problem

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

This paper 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. The PCA approach basically determines the components extracted from the data in the same space, i.e. the input space. However, in some cases it is useful to look at the components in a space of the original variables/features of the experiment, instead of the derived input space. Here the input space refers to the data we have in the current form, which might usually have been derived from a nonlinear transformation of variables' data in feature space. The dimensionality of the feature space might also be different from that of the input space. This paper develops a frame work to handle this task of performing a form of nonlinear PCA by using kernel function approach. The kernel functions are essentially the dot products in the feature space transformed by the non-linearity relating the input space to the feature space. The key feature of the Kernel PCA method is that we do not need to carry out the mapping of the data points into the feature space. All calculations can be done using the kernel function in the input space.

<u>Feature Space PCA</u>

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:

$$C = \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{x}_j \, \boldsymbol{x}_j^T$$

And solving the eigenvalue problem:

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

The detailed PCA algorithms is shown below:

Algorithm 1 PCA in Feature Spaces

```
1: procedure PCA(X)
2:     given input: $X_{n\times m} \leftarrow [\mathbf{x}_1; \mathbf{x}_2; \cdots ; \mathbf{x}_n]^T$
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: $E_{m\times k} \leftarrow [\mathbf{u}_1; \mathbf{u}_2; \cdots ; \mathbf{u}_k]$
7:     project the test data $\mathbf{x}$ : $\mathbf{p} \leftarrow E^T \mathbf{x}$
8: finish
```

Kernel PCA:

Now, suppose the non-linearity between input space and feature space is represented as a function Φ(). We then perform the PCA in the new space with the use of these transformations.

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

The covariance matrix in the feature space and the eigenvalue problem we solve in the feature space is:

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

The eigenvalue problem $\lambda \boldsymbol{V} = \bar{C}\boldsymbol{V}$ can also be expressed in terms of a dot product as follows:

$$\lambda(\Phi(x_k).\boldsymbol{V}) = (\Phi(x_k)\,.\bar{C}\boldsymbol{V}) \quad for\ all\ \ k = 1, \dots, M$$

We then obtain the normalized principal components in the new space.

Mercer's Theorem states that if k(**x**, **y**) is a continuous kernel of a positive integral operator, there exists a mapping into a space where k acts as a dot product. Dot product in new space is represented as a kernel so that the notation is simplified:

$$\mathrm{k}(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}).\Phi(\mathbf{y}))$$

In kernel PCA, a non-trivial arbitrary function Φ() is chosen, but is never calculated explicitly, allowing the possibility to use a very high dimensional Φ. Hence, we never actually compute the eigenvectors and eigenvalues of the covariance matrix in the high dimensional feature space. Elements in the K matrix represent the dot product of one point of the transformed data with respect to all the transformed points.

$$\boldsymbol{V}^n\,.\,\Phi(\boldsymbol{x}) = \sum_{i=1}^{M} \boldsymbol{\alpha}_i^n\, k(\boldsymbol{x}_i, \boldsymbol{x})$$

Where $\boldsymbol{\alpha}^n$ is the eigenvector of K, and $\lambda_1, \lambda_2 \ldots \lambda_M$, are the eigenvalues of K obtained by solving the equation $M\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}$. The algorithm is summarized below:

Algorithm 2 Kernel PCA

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

Properties of Kernel PCA:

The salient feature of the Kernel PCA is that when we are extracting the major modes/components from the data, we are not limited to extracting only up-to the dimensionality of our data set, but we can extract further components from the data. The usual properties of PCA continue to hold here, such as: The first q principal components (the projections of eigen vectors) 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.

Computational Complexity of the Kernel PCA:

Kernel PCA is not computationally intensive, because we are not transforming the data into feature space but we are computing the dot-products using the kernel function in the input space itself. Thus this procedure is equally computationally intensive as linear PCA. When there are large number of observations we estimate the Kernel Matrix using a subset of the input data to reduce the complexity of calculations.

Experiments by the Authors:

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

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 eigen values as the data is in 2 dimensions. In contrast the non-linear 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.

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 non-linear 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. Non-linear PCA addresses this issue better.

Advantages of Kernel PCA:

- In the use of pattern recognition, the components from nonlinear PCA recognized the features better than the linear PCA.
- We do not perform any non-linear optimization in this procedure, just solve an eigenvalue problem.
- We do not need to specify the number of principal components in advance to perform this method.

Experiments in This Project:

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

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.

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 x 16 greyscale 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.