De-noising and Recovering Images Based on Kernel PCA Theory

Pengcheng Xi
College of Information Science and Technology
Nanjing Univ. of Aeronautics & Astronautics
Nanjing, 210016 P.R. China
xipengchen@etang.com

Tao Xu
College of Information Science and Technology
Nanjing Univ. of Aeronautics & Astronautics
Nanjing, 210016 P.R. China
taoxucs@nuaa.edu.cn

ABSTRACT

Principal Component Analysis (PCA) is a basis transformation to diagonalize an estimate of the covariance matrix of input data and, the new coordinates in the Eigenvector basis are called principal components. Since Kernel PCA is just a PCA in feature space F, the projection of an image in input space can be reconstructed from its principal components in feature space. This enables us to perform several applications concerning de-noising and recovering images. Because of the superiority of Kernel PCA over linear PCA, we also get satisfactory effects of de-noising images using Kernel PCA.

Keywords

Kernel PCA; principal components; feature space; de-noising and recovering

1. INTRODUCTION

PCA is one of the most important techniques for feature extraction, and it has been widely used in fields such as de-noising and classification [1][6]. Concerning PCA, one can make use of a linear transformation to extract feature components. Kernel PCA, one of the nonlinear variants of PCA, is a generalization of linear PCA. Through a nonlinear map Φ, we can easily extract nonlinear features in feature space F via kernel functions k (x, y)= $\Phi(x) \bullet \Phi(y)$ rather than performing complicated calculations in input space [4].

By projecting data onto principal subspaces and thus dropping some components with small feature value, both linear PCA and Kernel PCA can reconstruct them with small variance.

2. PCA, Kernel PCA and de-noising

2.1 PCA

Principal Component Analysis (PCA) is a basis transformation to diagonalize an estimate of the covariance matrix of the data $x_k, k = 1, \ldots, l, x_k \in R^N, \sum_{k=1}^{l} x_k = 0$, and we define:

$$C = \frac{1}{l} \sum_{j=1}^{l} x_j x_j^T . \quad (1)$$

The new coordinates in the Eigenvector basis, that is the orthogonal projections onto the Eigenvectors, are called principal components [2].

In practical experiments, training set should be centered for processing:

$$C = \frac{1}{l} \sum_{i=1}^{l} (x_i - \mu)(x_i - \mu)^T . \quad (2)$$

where μ is the average image of input image set.

To eigen-decompose this C matrix is always an impractical task for high dimension of this matrix. Fortunately, alternative method using Singular Value Decomposition (SVD) is available to tackle this problem.

Concerning an image f, which is to be recognized or de-noised, we first get its projective vector:

$$y = U^T f \quad (3)$$

and then reconstruct it through:

$$\hat{f} = Uy \quad (4)$$

where *U* is the eigenvector matrix of C.

WSCG POSTERS proceedings
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

To do linear PCA, we only need to keep those eigenvectors in this matrix U corresponding to large eigen-value.

Next we generalize this process to a nonlinear one of the following kind. We first map the data in input space nonlinearly into a feature space F by

$$\Phi : R^N \to F, x \mapsto X. \qquad (5)$$

We know that mappings, which take us into a higher dimensional space than the dimension of the input space, provide us with greater classification power. However, the problem of high dimensionality is that this can seriously increase computation time. To tackle this problem, it is possible to take advantage of high dimensional representations without actually having to work in the high dimensional space [3].

2.2 Kernel PCA and de-noising

Let $\{ x_i \in R^N \}_{i=1}^{l}$ denote the input data. Kernel PCA first map the data into some feature space F, as is related in the above PCA descriptions. And then a standard PCA is performed on the mapped data.

Assume that our data mapped into feature space, $\Phi(x_1),...,\Phi(x_l)$, are centered. Then we can perform PCA on the covariance matrix

$$\bar{C} = \frac{1}{l}\sum_{j=1}^{l} \Phi(x_j)\Phi(x_j)^T, \qquad (6)$$

To find eigen-value $\lambda \geq 0$ and eigen-vector V, which satisfies $\lambda V = \bar{C}V$, we find an alternative method for this impractical computation because of rather high dimension. Accordingly we may consider the equivalent system

$$\lambda(\Phi(x_k) \bullet V)) = (\Phi(x_k) \bullet \bar{C}V) \text{ for all } k = 1,...,l, \qquad (7)$$

and there exist coefficients $\alpha_1,...,\alpha_l$ satisfying

$$V = \sum_{i-1}^{l} \alpha_i \Phi(x_i). \qquad (8)$$

Moreover we define a $l*l$ matrix K by $K_{ij} := (\Phi(x_i) \bullet \Phi(x_j)) = (k(x_i, x_j))$, then the above eigen-decomposition problem can be reduced to solve the following:

$$l\lambda\alpha = K\alpha \qquad (9)$$

where eigen-value λ, and eigen-vector $\alpha = (\alpha_1,...,\alpha_l)^T$.

Kernels include Gaussian kernel function:

$$k(x, y) = \exp(-\| x - y \|^2 / (2\sigma^2)), \qquad (10)$$

and polynomial kernel function $k(x, y) = (x \bullet y)^d$, which compute dot products in feature space.

For principal component extraction, we compute projections of the image of a test point $\Phi(x)$ onto the Eigen-vectors V^k in F according to

$$\beta_k := (V^k \bullet \Phi(x)) = \sum_{i=1}^{l} \alpha_i^k (\Phi(x_i) \bullet \Phi(x)) = \sum_{i=1}^{l} \alpha_i^k k(x_i, x). \qquad (11)$$

Thus we can perform PCA in the feature space whether the corresponding map Φ is known.

Because Kernel PCA is a PCA in F, the mapping data $\Phi(x)$ can be reconstructed from its principal components. To prove this, we first define an operator P_n (assuming that the eigenvectors are sorted so that the eigen-values decreased with k)

$$P_n \Phi(x) = \sum_{k=1}^{n} \beta_k V^k. \qquad (12)$$

We would like to find approximate representations of the data in input space rather than in F. Thus we are looking for a $z \in R^N$ so that

$$\sigma(z) = \| P_n \Phi(x) - \Phi(z) \|^2 \qquad (13)$$

2.3 Iterative De-noising with Gaussian Kernels

In order to perform de-noising, we need to reconstruct mapped data in input space R^N rather than in feature space F.

Given x in input space, we first map it into $\Phi(x)$, drop those components corresponding to small eigen-value to achieve $P_n\Phi(x)$, and then compute z. Such computation is at the aim of capturing the main structure of the data set in the first n directions, and the remaining components just pick up the noise. Thus z can be regarded a de-noised version of x.

In practice, the equation of $P_n\Phi(x) = \Phi(x)$ cannot be well satisfied. Therefore, the existence of an exact pre-image of a mapped one may not be guaranteed. Under such circumstances, we use gradient descent methods to minimize (13) over z. Substituting (8) and (12) into (13), we can reduce the above objective expression to the following:

$$\sigma(z) = C - \sum_{k=1}^{n} \beta_k \sum_{i=1}^{l} \alpha_i^k k(x_i, z) \qquad (14)$$

here C denotes a common value independent of z [3].

To tackle the same problem, Mika et. al [5] also proposed another method to approximate z by minimizing $\sigma(z) = \| P_n\Phi(x) - \Phi(z) \|^2$. Since

kernels satisfying $k(x,x) \equiv cons$ for all x, we can maximize the following expression instead of $\sigma(z)$.

$$\tilde{\sigma}(z) = (\Phi(z) \bullet P_n \Phi(x)) + C = \sum_{i=1}^{l} \alpha_i k(x_i, z) + C \qquad (15)$$

3. Experiments

Our first experiment is based on the training set composed of 12 images of a Chinese character of different fonts (Fig 1).

Fig 1: Set of training images

From this training set, we drew one image, to which we added Gaussian noise (37.5%) as the one to be de-noised. Furthermore, upon this noised image, we re-added Gaussian noise (also 37.5%) as the starting point of z in equation (14). Fig 2 shows the noised image (left) and re-noised one (right) respectively.

Fig 2: Testing & starting images

Fig 3 gives a comparison of de-noising effect with different Gaussian factors and principal components (with each team of four images' comparison, the principal components are correspondingly 11, 9, 4, 1 from left to right).

$K(x,y) = \exp(-\| x-y \|^2 / 3000)$

$K(x,y) = \exp(-\| x-y \|^2 / 1000)$

$K(x,y) = \exp(-\| x-y \|^2 / 600)$

$K(x,y) = \exp(-\| x-y \|^2 / 300)$

Fig 3: Effect of de-noising for different Gaussian factors and principal components

During the process of iteration, another factor, which may contribute to the velocity of processing, is the pace we chose. Concerning this problem, narrow pace may result in too many times of iteration, while improperly large pace may lead to no results. Thus iteration times are not suitable for acting as the criteria of comparison.

Despite these details, we easily draw from the above comparison that, with the decrement of principal components, there is an increment of background noise. Therefore, the indication that principal components as many as possible are able to maintain main constructions of an image is well proved here. In addition, we find that, concerning particular image set, there is a proper Gaussian factor, which can contribute to the best effect of de-noising and reconstructing images.

Another experiment is based on the same training set as mentioned above; however, image to be de-noised is replaced by a noised handwritten character. Here two similar examples of different fonts are shown in Fig 4.

Fig4: Noised handwritten image and de-noised one

From the above two experiments concerning handwritten images, we can learn the possibility that we may not find an exact pre-image of that noised one; nevertheless, we may find the original image in the training set, which is most closest to the noised one.

To prove that our method also works well on de-noising gray level images, another experiment is performed here on another training set, in which images are different for their various gray level distributions (Fig 5). Similarly an image is picked out from the training set to add noise as the one to be de-noised, and the effect of de-noising is shown in Fig 6.

From the experiment above, we successfully tested the quality of Kernel PCA method on gray level

images. Although the images we picked here are of simple gray level distributions, we can optimistically predict that our method also performs well in de-noising more complicated gray level images.

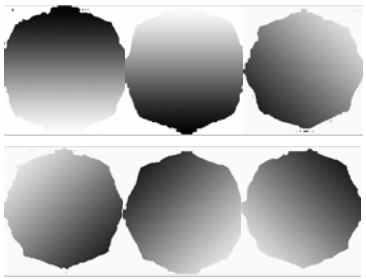

Fig 5: Set of gray level images

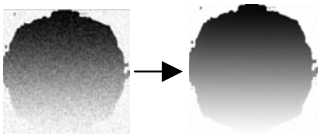

Fig 6: Noised gray level image and de-noised one

To prove that our method out-performs linear PCA in de-noising and reconstructing images, we did another experiment with linear PCA based on training set of Fig 1. From the comparison shown in Fig 7, we find that although the Gaussian noise we added was wiped out, the background was simultaneously added with new noise.

Fig 7: De-noising images by linear PCA (first is noised image, next four with principal components of 11, 9, 4, 1 respectively)

Here we use the mean square error (MSE) as a criterion for comparing the effect of de-noising between our method and linear PCA (Table 1).

Principle components	11	9	4	1
Linear PCA	85.2383	85.2891	94.1005	94.3518
Kernel PCA	0.0036	0.0040	0.0051	0.0059

Table 1: Comparison of mean square error between linear PCA and Kernel PCA

This comparison gives us a clear impression that the Kernel PCA out-performs linear PCA in de-noising and reconstructing images. Since the new background noise is added as mentioned above, the difference between MSE of PCA and Kernel PCA is very significant.

4. Conclusion

Here we have studied the problem of finding pre-image of vectors in feature spaces, and shown how it is related to the problem of reconstructing data from its nonlinear principal components as extracted by the Kernel PCA algorithm. Through experiments on de-noising different kinds of images and comparison between Kernel PCA and linear PCA, we find some interesting conclusions, which include that as many as possible of principal components can keep main images and that, Kernel PCA outmaneuvers linear PCA in the application of de-noising and reconstruction.

References

[1] Takashi Takahashi and Takio Kurita. Robust De-noising by Kernel PCA. International Conference on Artificial Neural Networks (ICANN2002), In Artificial Neural Networks - ICANN2002, Springer, pp. 739-744, 2002

[2] B. Scholkopf, A. J. Smola, and K.R. Muller. Kernel Principal Component Analysis. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, advances in Kernel Methods - SV Learning, pp. 327-352. MIT Press, Cambridge, MA,1999.

[3] B. Scholkopf, S. Mika, A. Smola, G. Ratsch, K.R. Muller. Kernel PCA Pattern Reconstruction via Approximate Pre-Images, In: Niklasson L., Boden M. and Ziemke T. (eds.), Proceedings of the 8th International Conference on Artificial Neural Networks, Springer Verlag, Perspectives in Neural Computing, pp. 147-152, 1998.

[4] B. Scholkopf, A. J. Smola, and K.R. Muller. Nonlinear component analysis as a kernel eigen-value problem. Neural Computation, 10: 1299-1319, 1998.

[5] B. Scholkopf, S. Mika, C.J.C. Burges, P. Knirsch, K. –R. Muller, G. Raetsch and A. Smola. Input space vs. feature space in kernel-based methods. IEEE Transactions on Neural Networks, 1999, 10:5, pp. 1000-1017.

[6] A. M. Jade, B. Srikanth, V.K. Jayaraman, B. D. Kulkarni, J.P. Jog, L. Priya. Feature extraction and de-noising using kernel PCA. Chemical Engineering Science 58 (2003) 4441-4448.