Highlighted Technical Paper A

Generalization of Kernel PCA and Automatic Parameter Tuning

Takahide Nogayama, Haruhisa Takahashi, Masakazu Muramatsu
The University of Electro-Communications
Chofugaoka 1–5–1, Chofu-shi, Tokyo, 182-8585, Japan
noga@ice.uec.ac.jp

Abstract

We propose a generalized kernel PCA which provides much more accurate information of kernel space. Calculating partial derivatives of eigenvalues with kernel parameters, we can obtain the optimal kernel parameters. The criterion for optimal parameters are given by a quadratic cost function with respect to eigenvalues. We compared our method with SVM for face recognition, and showed that our method works efficiently as expected.

1. Introduction

Kernel based methods work by mapping data into a high dimensional feature space defined by the kernel function that computes inner product of two input images. There are several related methods such as support vector machine (SVM)[1], kernel principal component analysis (KPCA)[7] [6], and kernel Fisher's discriminant analysis (KFDA)[3].

Kernel based methods potentially provide a proficient approach to pattern recognition. KPCA proposed by Schölkopf [7] [6] can extract nonlinear component by performing PCA on the kernel feature space. It is expected for effective application to pattern recognition, and recently draws intensive attention in the face recognition society [9] [8].

A particular application generally requires sensible choice of a specific kernel function, and in pattern recognition the classification performance largely depends on this choice. In some application such as face recognition, radial basis function is in popular use, and so the choice of kernel function reduces to the problem of choosing of kernel parameters. If the parameters are appropriately chosen, one can extract optimal performance. Unfortunately there have been no effective methods so far for this issue.

Usually the parameters are determined empirically, or with auxiliary experiments, but in some cases such as multi-class problems and multi-parameter cases, it would be unreasonable for huge computational effort. In this paper we generalize Schölkopf's method and apply it to automatic determination of semi-optimal parameters for Gaussian kernel. The method makes use of the derivatives of kernel matrix eigenvalues to obtain iterative methods for optimal values using.

The following notation of the gaussian kernel is used through this paper

$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\right) \tag{1}$$

where σ is called kernel parameter.

2. Generalization of kernel PCA

As was mentioned in Introduction, KPCA proposed by Schölkopf performs PCA for features on the kernel space. PCA usually put the median point at the average vector of features. Since Schölkopf performed the analysis assuming that the fiducial point is at the origin in the kernel feature space, but it does not reflect the true distribution of variances in the kernel space. In this section we review his analysis setting the fiducial point at the average vector of features. We call this generalized KPCA.

2.1. Kernel method

Let $\boldsymbol{x} = (x_1, \ldots, x_n)^t$ be an observed feature vector, and let the set of vectors be $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\}$. We assume that $\boldsymbol{x}$ is mapped to p dimensional kernel feature space F by $\boldsymbol{\phi}(\boldsymbol{x}) \in F$.

$$\boldsymbol{x} = (x_1, \ldots, x_n)^t \mapsto \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_p(\boldsymbol{x}))^t \tag{2}$$

Kernel method computes inner product

$$k(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{z}) \rangle = \boldsymbol{\phi}(\boldsymbol{x})^t \boldsymbol{\phi}(\boldsymbol{z}) \tag{3}$$

without referring the feature ϕ_i $(i = 1, \ldots, p)$ directly. Let K be $K_{ij} = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, then $K = \Phi^t \Phi$ with $\Phi = (\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_l))$ (sample matrix).

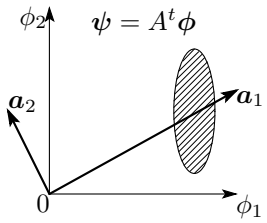

Figure 1. KPCA

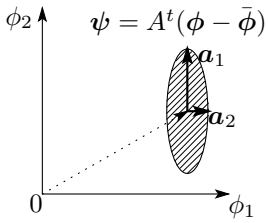

Figure 2. GKPCA

2.2. Kernel PCA with fiducial point at the mean vector

There are two methods for determining principal component axis : variance maximization criterion and mean square error minimization criterion. In this paper we adopt the former criterion. This criterion maximize the variances in feature space which is linearly transformed by some orthonormal system.

Performing PCA in kernel space, we assume that p dimensional linear space is transformed to d dimensional linear space. Let $\boldsymbol{a}_i$ be an orthonormal vector of the d-dimensional linear sub-space, and let $A = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_d)$. We assume $\boldsymbol{\phi}(\boldsymbol{x})$ is transformed to $\boldsymbol{\psi}^t(\boldsymbol{x}) = (\psi_1(\boldsymbol{x}), \ldots, \psi_d(\boldsymbol{x}))$ by $\boldsymbol{a}$ $(i = 1, \ldots, l)$, where

$$\boldsymbol{\psi}(\boldsymbol{x}) = A^t\left(\boldsymbol{\phi}(\boldsymbol{x}) - \bar{\boldsymbol{\phi}}\right) \tag{4}$$

and where $\bar{\boldsymbol{\phi}} = \frac{1}{l}\sum_{i=1}^{l}\boldsymbol{\phi}(\boldsymbol{x}_i)$ (see Figure 2). Let $\mathbf{1}$ be the $l \times 1$ vector, where all entries equal unity. Now let $p \times p$ covariance matrix in kernel space be Σ_ϕ. Then

$$\Sigma_\phi = \frac{1}{l}\Phi\left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)\left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)\Phi^t. \tag{5}$$

Σ_ψ is $d \times d$ covariance matrix in the d-dimensional subspace given by eq. (4) or

$$\Sigma_\psi = A^t\Sigma_\phi A. \tag{6}$$

Because orthonormal bases $\boldsymbol{a}_i$ $(i = 1, \ldots, d)$ that we want to obtain are in the kernel space, we can represent them as a linear combination of $\left(\boldsymbol{\phi}(\boldsymbol{x}_i) - \bar{\boldsymbol{\phi}}\right) \in F$, $(i = 1, \ldots, l)$. Assuming $\boldsymbol{b}_i = (b_{i1}, \ldots, b_{il})^t$ be the coefficient vectors

$$\boldsymbol{a}_i = \sum_{j=1}^{l} b_{ij}\left(\boldsymbol{\phi}(\boldsymbol{x}_j) - \bar{\boldsymbol{\phi}}\right). \tag{7}$$

Further $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$

$$A = \Phi\left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)B. \tag{8}$$

Now we can formalize PCA as an optimization problem. Let $G = \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)\Phi^t\Phi\left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)$. Then maximizing sum of variances in subspace using orthonormal bases is

$$\text{maximize} \quad : \quad tr\left(\Sigma_\psi\right) = \frac{1}{l}tr\left(B^tG^2B\right) \tag{9}$$

$$\text{subject to} \quad : \quad A^tA = B^tGB = I_d. \tag{10}$$

To obtain the solution, We can use Lagrangian function

$$L(B, \Lambda) = \frac{1}{l}tr\left(B^tG^2B\right) - tr\left(\left(B^tGB - I_d\right)\Lambda\right). \tag{11}$$

Then setting

$$\frac{\partial L}{\partial B} = \frac{2}{l}G^2B - 2GB\Lambda \tag{12}$$

be zero, we obtain Karush-Kuhn-Tucker(KKT) condition

$$\frac{1}{l}G^2B = GB\Lambda \tag{13}$$

$$B^tGB = I_d. \tag{14}$$

Solving this KKT condition for B, we can obtain the solution.

Let the ith eigenvalue of G be μ_i with corresponding eigenvector $\boldsymbol{v}_i$. Assume that $\boldsymbol{v}_i^t\boldsymbol{v}_i = 1$ $(i = 1, \ldots, l)$, and $\mu_1 \geq \ldots \geq \mu_d \geq 0$. Since G is a symmetric matrix, eigenvectors constitute orthonormal set. Thus letting $\boldsymbol{b}_i = \frac{1}{\sqrt{\mu_i}}\boldsymbol{v}_i$, we obtain $\boldsymbol{b}_i^tG\boldsymbol{b}_j = \delta_{ij}$ which satisfy eq. (14). We denote $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_l)$, the diagonal matrix M as $M_{ii} = \mu_i$, and $M^{-\frac{1}{2}}$ as

$$[M^{-\frac{1}{2}}]_{ij} = \begin{cases} \frac{1}{\sqrt{\mu_i}} & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}. \tag{15}$$

Then

$$B = VM^{-\frac{1}{2}}. \tag{16}$$

And eq. (13) can be transformed to

$$\text{(left side)} \quad = \quad \frac{1}{l}G^2B = \frac{1}{l}VM^2M^{-\frac{1}{2}} \tag{17}$$

$$\text{(right side)} \quad = \quad GB\Lambda = VMM^{-\frac{1}{2}}\Lambda \tag{18}$$

by using $GV = VM$. Since V, M are non zero matrix ,

$$VM\left(\frac{1}{l}M - \Lambda\right)M^{-\frac{1}{2}} = O \tag{19}$$

results in $\Lambda = \frac{1}{l}M$ or $\lambda_i = \frac{\mu_i}{l}$ $(i = 1, \ldots, d)$.

One can easily check $\Sigma_\psi = \Lambda$ is substituting B and Λ into Σ_ψ and using eq.(6) , eq.(8). This means the variance on ith axis in the d-dimensional subspace is λ_i, and each axes are uncorrelated.

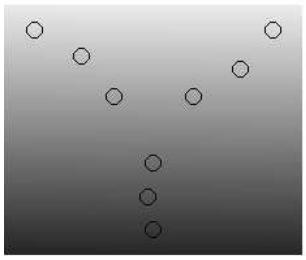

Figure 3. Linear PCA 1st principal component

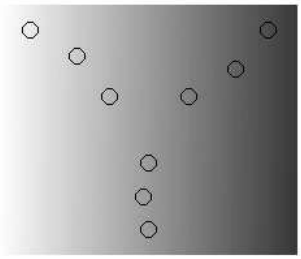

Figure 4. Linear PCA 2nd principal component

Finally $\psi(\boldsymbol{z})$ is represented by

$$\begin{aligned}\psi(\boldsymbol{z}) &= A^t\left(\phi(\boldsymbol{z}) - \bar{\phi}\right) \\ &= B^t\left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)\Phi\left(\phi(\boldsymbol{z}) - \frac{1}{l}\Phi\mathbf{1}\right) \\ &= M^{-\frac{1}{2}}V^t\left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)\left(\boldsymbol{k}(\boldsymbol{z}) - \frac{1}{l}K\mathbf{1}\right) \quad (20)\end{aligned}$$

using $\boldsymbol{k}(\boldsymbol{z}) = (K(\boldsymbol{x}_1, \boldsymbol{z}), \ldots, K(\boldsymbol{x}_l, \boldsymbol{z}))^t$.

2.3. Illustration via artificial data

Now we illustrate the generalized KPCA using artificial data. Consider a data of Figure 3 and 4 which shows stellately distributed data on two dimensional space having three actinoid wings.

If we apply linear principal component analysis for this data, we obtain only two principal components which are clearly unable to represent the data. The results are shown in Figure 3,4. Thus KPCA would be an efficient tool in this example.

Figure 5 shows the result of applying the generalized KPCA for this data. In this example one can see that a small parameter can give only features neighbor of the data, and large parameter is too crude to represent precise features. In a later section we will see that the optimum parameter in this case is $\sigma = e^{0.7}$.

In Figure's 6, 7 the landscapes of variance for each principal component axis scanned with kernel parameter are shown for KPCA and the generalized KPCA (GKPCA), respectively. From Figure 7 (GKPCA), one can see that small parameters give equal variance for each axis, thus each component is equally important. On the other hand the large parameters give zero variance for each axis meaning the distribution of data is concentrated on data points. KPCA (Figure 6), on the other hand, can not represent this, because the center point is not appropriately set.

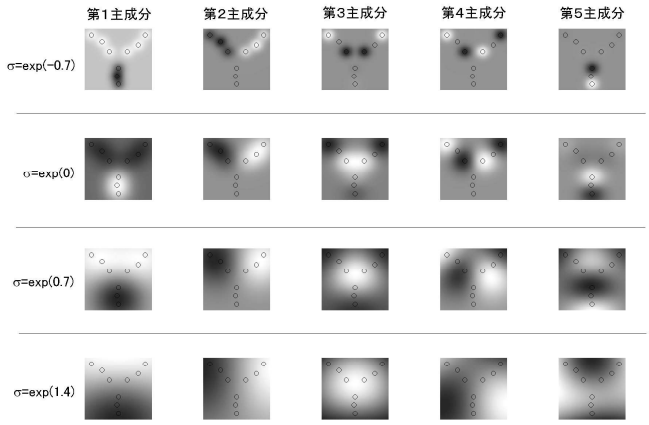

Figure 5. Kernel parameter and principal component of GKPCA

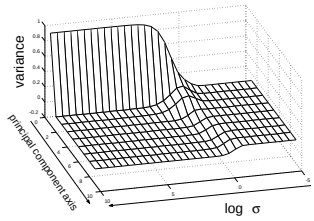

Figure 6. Variance vs. principal component and kernel parameter (KPCA)

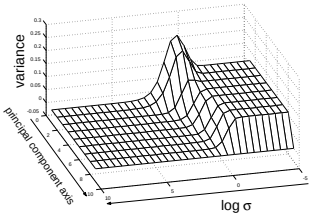

Figure 7. Variance vs. principal component and kernel parameter(GKPCA)

3. Automatic parameter tuning

As was mentioned in the previous section, when the kernel parameter is too large, each axis in the kernel space has similar variance. On the contrary when the kernel parameter is too small, all axes have zero variances losing data information. The optimal parameter should be in the middle of these extremes. Thus the optimal kernel space should consist of a few principal component axes with large variance and remaining large number of principal component axes with small variance. This is illustrated in Figure 8. We require that the optimal kernel parameter should be maximizing the deviation of variances distribution.

Let kernel parameter be θ, and let variance of ith principal component axis be λ_i. According to our request, optimal kernel parameter maximizes following cost function (see Figure 8).

$$E(\theta) \quad = \quad \frac{1}{l}\sum_{i=1}^{l}\left(\lambda_i - \bar{\lambda}\right)^2$$

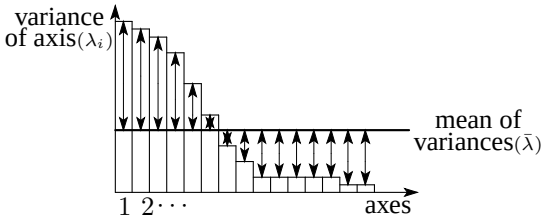

Figure 8. Deviation of variances

$$= \frac{1}{l}\boldsymbol{\lambda}^t \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t \right) \boldsymbol{\lambda} \tag{21}$$

Eq.(21) can be optimized by the following steepest descent gradient method:

$$\theta_{t+1} = \theta_t + \eta \frac{\partial E}{\partial \theta}. \tag{22}$$

In eq.(23), the derivative of cost function with θ is given by

$$\frac{\partial E}{\partial \theta} = \frac{2}{l}\boldsymbol{\lambda} \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t \right) \frac{\partial \boldsymbol{\lambda}}{\partial \theta} \tag{23}$$

$$= \frac{2}{l^3}\boldsymbol{\mu} \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t \right) \frac{\partial \boldsymbol{\mu}}{\partial \theta}. \tag{24}$$

To obtain $\frac{\partial \mu_k}{\partial \theta}$, we apply the implicit function theorem and the derivation of determinant. Since μ_k is an eigenvalue of $G(\theta)$, it satisfies characteristic equation $\det\left(\mu_k I_l - G\left(\theta\right)\right) = 0$. Thus there exists implicit function $f_{\mu_k}\left(\cdot\right)$ such as $f_{\mu_k}\left(\theta\right) = \mu_k$. Let C be the transposed cofactor matrix of $\mu_k I_l - G$. Then if $\sum_{i=1}^{l} C_{ii} \neq 0$,

$$\frac{\partial f_{\mu_k}}{\partial \theta} = -\frac{\frac{\partial \det(\mu_k I_l - G)}{\partial \theta}}{\frac{\partial \det(\mu_k I_l - G)}{\partial \mu_k}} \tag{25}$$

$$= \frac{\sum_{i=1}^{l}\sum_{j=1}^{l} C_{ij}\frac{\partial G_{ij}}{\partial \theta}}{\sum_{i=1}^{l} C_{ii}}. \tag{26}$$

To avoid much computational effort for calculating the transposed cofactor matrix, we refer the following theorem without proof.

Theorem 1 *Let eigenvalues and eigenvectors of G be $\mu_i, \boldsymbol{v}_i$ $(i = 1, \ldots, l)$, respectively. Then transposed cofactor matrix of $(\mu_k I_l - G)$ is given by*

$$\prod_{\substack{i=1 \\ i\neq k}}^{l} \left(\mu_i - \mu_k\right) \boldsymbol{v}_k \boldsymbol{v}_k^t. \tag{27}$$

This theorem can be proved by a regularization method of a cofactor matrix and inverse matrix.

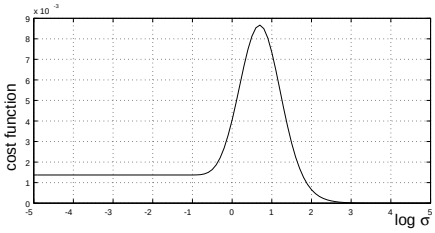

Figure 9. Cost function

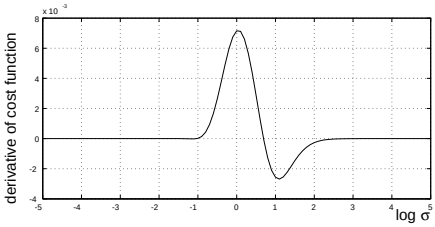

Figure 10. Derivative of cost function

Using Theorem 1, $\frac{\partial \mu_k}{\partial \theta}$ is calculated

$$\frac{\partial f_{\mu_k}}{\partial \theta} = \frac{\prod_{\substack{i=1 \\ i\neq k}}^{l}\left(\mu_i - \mu_k\right)\sum_{i=1}^{l}\sum_{j=1}^{l}(\boldsymbol{v}_k \boldsymbol{v}_k^t)_{ij}\frac{\partial G_{ij}}{\partial \theta}}{\prod_{\substack{i=1 \\ i\neq k}}^{l}\left(\mu_i - \mu_k\right)\sum_{i=1}^{l}[\boldsymbol{v}\boldsymbol{v}^t]_{ii}}$$

$$= \frac{\prod_{\substack{i=1 \\ i\neq k}}^{l}\left(\mu_i - \mu_k\right)\boldsymbol{v}_k^t \frac{\partial G}{\partial \theta}\boldsymbol{v}_k}{\prod_{\substack{i=1 \\ i\neq k}}^{l}\left(\mu_i - \mu_k\right)}$$

$$= \boldsymbol{v}_k^t \frac{\partial G}{\partial \theta}\boldsymbol{v}_k. \tag{28}$$

The following Learning algorithm gives the optimizing procedure.

- $t = 1, \quad \theta_t := \theta_0$
- repeat
  - $G := \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right) K(\theta_t) \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)$
  - $\frac{\partial G}{\partial \theta} := \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right) \frac{\partial K(\theta_t)}{\partial \theta} \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right)$
  - $\boldsymbol{\mu} :=$ eigenvalues of G
  - $V :=$ eigenvectors of G
  - for $i := 1$ to l
    - $\frac{\partial \mu_i}{\partial \theta} = \boldsymbol{v}_i^t \frac{\partial G}{\partial \theta}\boldsymbol{v}_i$
  - $\frac{\partial E}{\partial \theta} := \frac{1}{l^3}\boldsymbol{\mu}^t \left(I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^t\right) \frac{\partial \boldsymbol{\mu}}{\partial \theta}$
  - $\theta_{t+1} := \theta_t + \eta \frac{\partial E}{\partial \theta}$
- until $(\theta_{t+1} \simeq \theta_t)$

Figure 9, 10 illustrate the cost function and its derivative as functions of the kernel parameter for the data used in section 2.3. One can easily see that the cost function gives its maximum at $\sigma = e^{0.7}$ where the derivative is zero.

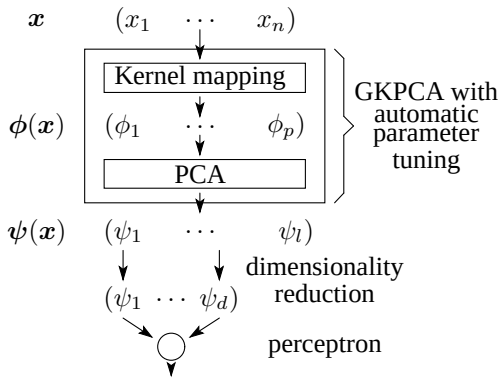

Figure 11. GKPCA-plus-perceptron

4. Experimental results

4.1. Learning with GKPCA-plus-perceptron

Since PCA is the preprocessing step for classification task, we use perceptron(single layer) as a classifier. Because GKPCA prepare powerful nonlinear mapping and enabling appropriate dimensionality reduction, such simple classifier is sufficient for classification tasks. To enable to learn weakly nonlinearly separable problem, we employ the sigmoidal function for the output function.

$$f(u) = \frac{1}{1+\exp(-u)} \quad , \quad \left(u = \boldsymbol{w}^t \boldsymbol{x} + b\right) \tag{29}$$

Perceptron has only one parameter called learning rate η. Since perceptron learning approximately converges to the same solution without regard to a value of the learning rate, it has no heuristic parameters. Optimizing the kernel parameter by our method, "GKPCA-plus-perceptron" also has no heuristic parameter. (see Figure 11).

4.2. Face recognition

To test our method, we applied it to face images of the HOIP database [1] . The data consist of 300 images of 150 males and 150 females, and each image is normalized to 46×44 pixels having real number from 0 through 1. Thus the input vector has 2024-dimensions. We classified male and female into two categories (see Figure 12). We performed 10-fold-cross-varidation with randomly divided data, and took average of five 10-fold-cross-validations. Perceptron's learning rate was set to $\eta = 0.1$. The landscape of variance for each principal component axis scanned with kernel parameter σ are shown in Figure 13.

[1]The facial data in this paper are used by permission of Softopia Japan, Research and Development Division, HOIP Laboratory. It is strictly prohibited to copy, use, or distribute the facial data without permission.

Figure 12. Face images used in experiment

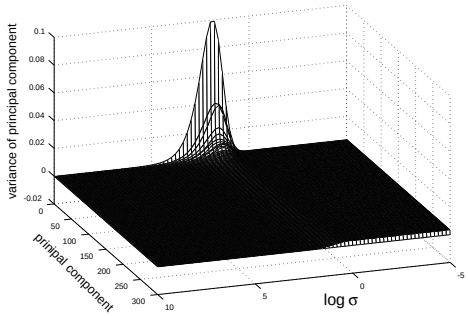

Figure 13. Variances scanned with kernel parameter

In Figure 14, 15, the training error and the testing error are shown for the parameter and the number of principal components. The experiments were done for each parameter σ, changing the number of principal components (Note that principal components are sorted in descending order). Optimal parameter determined by our method is near $\sigma = e^{1.8}$ in this case (see Figure 13). One can observe that both errors become small near the optimal parameter, although we use only few principal component axes (see Figure 16). The minimal value of average testing error was 7.53% for the optimal parameter and for the number of principal component $d = 155$. The true minimal value of average testing error was 6.13% obtained for the parameter scan with $\sigma = [e^{-5}, e^{-4.8}, \ldots, e^{9.8}, e^{10}]$ for $d = 155$ (see Figure 15, 16). Both errors were averaged over 50 times simulation-runs.

The optimal parameter was computed by eq.(22) setting $\eta = 10000$, and setting σ_0 randomly from [1.0, 20.0]. The average computational time for the optimal solution is 8.01 sec. [2]

We compared our method with support vector machine using gaussian kernel. SMO algorithm [5] was used for training SVM. We performed exhaustive search for the optimal parameters σ and C, setting $\sigma = [e^{-5}, e^{-4}, \ldots, e^{10}]$, $C = [e^{-5}, e^{-4}, \ldots, e^{10}]$. Note that to find out the optimal parameter, we need to take an average of several experiments, e.g. of 50 times. The average computational time for the search was 39.58 sec / experiment, and totally it takes

[2]The programs were run on Pentium4(2.53GHz) with 512MB RAM.

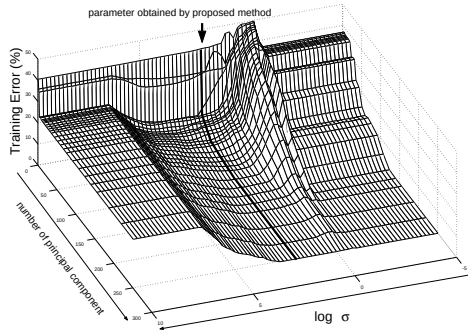

Figure 14. Average training error

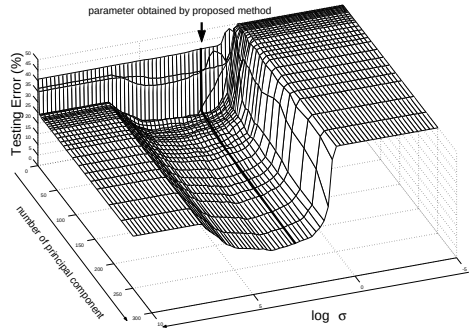

Figure 15. Average testing error

1979 sec to obtain the optimal kernel parameters. The minimal value of average testing error is obtained as 5.86% with $\sigma = e^5, C = e^7$.

We conclude from the experiments that the kernel parameter is near optimal. Although, we employ the perceptron algorithm, generalization errors are different only 2% from SVM.

We characterized the relationship between kernel parameter and eigenvalues about the data sets (iris, wine, glass, baberman, bupa-liver-disorders, ionosphere, breast-cancer-wisconsin, vehicle, vowel, segment) collected from UCI Repository of machine-learning-databases(http:// www.ics.uci.edu /˜mlearn /MLRepository.html). Therefore the data set which has distribution of eigenvalues similar to Figure 7 or Figure 13 seems to avoid the local minimum problem with our optimization method. All of the data sets indicates that each distribution of the eigenvalues resembles the used face data.

5. Conclusion

In this paper, we proposed a method which can tune the kernel parameter automatically. Furthermore the merit applying GKPCA-plus-perceptron is that we can solve any classification problem without heuristically determined parameters. We showed GKPCA-plus-perceptron can attain enough classification performance which is comparable to SVM.

The extention to the multi-class problem would have some merits because each class can have different parameter by our method.

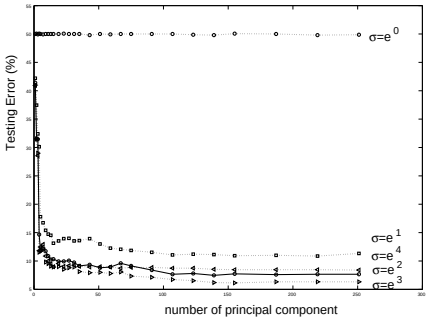

Figure 16. Average testing error

6. Acknowledgements

The helpful comments of the Prof. Kazuhiro Hotta on preliminary versions of the article are gratefully acknowledged. and the helpful proofreading of Mr. Yasuhide Miura is also gratefully acknowledged.

References

[1] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and othe kernel-based learning methods*. Cambridge University Press, 2000.

[2] G. F. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory*, IT-14:55–63, 1968.

[3] S. Mika, G. Ratsh, J. Weston, B. Scholkopf, and K.-R. Muller. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing IX*, pages 41–48, 1999.

[4] R. O.Duda, P. E.Hart, and D. G.stork. *Pattern Classification Second Edition*. John Wiley & Sons,. Inc, 2000.

[5] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods*, pages 185–208, Dec 1998.

[6] B. Scholkopf, A. J. Smola, and K.-R. Muller. Kernel principal component analysis. *Advances in Kernel Methods*, pages 327–353, Dec 1998.

[7] B. Scholkopf, A. J. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, July 1998.

[8] T. Takahashi and T. Kurita. Robust de-noising by kernel pca. *Lecture Notes in Computer Science*, pages 739–744, 2002.

[9] M.-H. Yang. Kernel eigenfaces vs. kernel fi shrefaces: Face recognition using kernel methods. *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 215–220, May 2002.