Neurocomputing 73 (2010) 1694–1703

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Supervised learning of local projection kernels

Mehmet Gönen *, Ethem Alpaydın

Department of Computer Engineering, Boğaziçi University, TR-34342 Bebek, İstanbul, Turkey

ARTICLE INFO

Available online 12 March 2010

Keywords:
Dimensionality reduction
Local embedding
Kernel machines
Subspace learning

ABSTRACT

We formulate a supervised, localized dimensionality reduction method using a gating model that divides up the input space into regions and selects the dimensionality reduction projection separately in each region. The gating model, the locally linear projections, and the kernel-based supervised learning algorithm which uses them in its kernels are coupled and their training is performed with an alternating optimization procedure. Our proposed *local projection kernel* projects a data instance into different feature spaces by using the local projection matrices, combines them with the gating model, and performs the dot product in the combined feature space. Empirical results on benchmark data sets for visualization and classification tasks validate the idea. The method is generalizable to regression estimation and novelty detection.

1. Introduction

In binary classification, we are given a training data set $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^{n}$ where n is the number of training instances, $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1,+1\}$. Dimensionality reduction is commonly used to alleviate the effect of redundant or correlated features and to visualize the training data using few, for example, two dimensions.

Principal component analysis (PCA) is the first method that comes to mind for linear dimensionality reduction. PCA seeks to maximize the explained variance of the data in the projected feature space and performs a linear dimensionality reduction by calculating a projection matrix from the eigenvectors of the covariance matrix. It may perform badly for classification problems due to its linear and unsupervised nature. Kernel PCA is an extension to PCA algorithm which obtains nonlinear mappings with the help of kernel functions [1].

Fisher discriminant analysis (FDA) is a well-known linear supervised method for dimensionality reduction that jointly minimizes the within-class variance and maximizes the between-class variance. FDA has two main limitations: (a) the dimensionality of the projected feature space can be at most $c-1$ where c is the number of classes, (b) it assumes that each class follows a unimodal distribution, which may not always hold. FDA can also be kernelized to obtain nonlinear mappings.

Methods such as PCA and FDA learn a global projection matrix and use this matrix over the whole input space. This approach may not work for data sets which have a local neighborhood structure. A mixture of principal component analyzers has been proposed to capture regional differences in the covariance structure [2]. The method divides the input density into clusters and learns a local PCA model in each cluster. However, the unsupervised nature of PCA method is preserved even though we learn local models.

Locally linear embedding (LLE) [3], Isomap [4], and Laplacian eigenmaps [5] are some examples of unsupervised locality preserving manifold learning algorithms but these methods do not explicitly learn a mapping function for unseen data instances. Onclinx et al. [6] and Hou et al. [7] propose two variants of the LLE algorithm in order to capture the local neighborhood structure in the data better. *Locality preserving projections* (LPP) have been proposed also to learn a mapping function while preserving locality [8].

In addition to these unsupervised methods, there are also supervised methods which keep the local neighborhood structure in the data. *Local Fisher discriminant analysis* (LFDA) combines the ideas behind FDA and LPP [9]. The mapping is obtained again by solving a generalized eigenvalue problem but the between-class scatter and within-class scatter matrices are calculated locally with the help of an affinity matrix (an idea which is borrowed from LPP). LFDA also removes the restriction of obtaining at most $c-1$ dimensions in the projected feature space. Tao et al. [10] extend FDA by maximizing the geometric mean of the divergences between different pairs of classes. This strategy obtains better projections in terms of class separation for multiclass problems. Another method is *local learning projections* (LLP) which can use supervised information (its difference from PCA) and minimize the local estimation error instead of the global estimation error [11].

* Corresponding author.
E-mail addresses: gonen@boun.edu.tr (M. Gönen), alpaydin@boun.edu.tr (E. Alpaydın).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.11.043

Yan et al. [12] formulate a common framework for representing different dimensionality reduction algorithms as graph embedding problems. For example, PCA, FDA, Isomap, LLE, LPP, and Laplacian eigenmaps can be cast into a common formulation. Following this idea, a supervised variant of LLE called discriminant LLE that also learns a mapping function is proposed [13].

In this paper, we propose a supervised dimensionality reduction method coupled with a kernel machine called *local projection kernels* (LPK). In Section 2, we reproduce the modification of the discriminant function of the *support vector machine* (SVM) by integrating a projection matrix and explain how to optimize SVM parameters and the projection matrix jointly, as given by Chapelle et al. [14]. We give a brief description of localized kernel functions proposed by Gönen and Alpaydın [15] in Section 3. Then, in Section 4, we combine these two ideas of projections and localized kernels, and describe how to optimize all of the parameters in a coupled manner with an alternating optimization procedure. Section 5 explains the key properties of the proposed algorithm. We then demonstrate its performance on benchmark data sets for visualization and classification tasks in Section 6 and conclude in Section 7.

2. Supervised learning of projection kernels

SVM is a discriminative classifier based on structural risk minimization [16] and soft-margin SVM formulation can be given as

$$\begin{aligned} &\text{min.} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{w.r.t.} \quad \boldsymbol{w},b,\boldsymbol{\xi} \\ &\text{s.t.} \quad y_i(\langle \boldsymbol{w},\Phi(\boldsymbol{x}_i)\rangle + b) \geq 1-\xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i \end{aligned} \tag{1}$$

where $\boldsymbol{w}$ is the vector of weight coefficients, C is the regularization parameter, b is the threshold, $\boldsymbol{\xi}$ is the vector of slack variables, and $\Phi(\cdot)$ is the mapping function used for classification. Suppose that, instead of using the original features, we apply a linear projection to data instances with the projection matrix, $\boldsymbol{W} \in \mathbb{R}^{d\times r}$:

$$\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{x}$$

where r is the dimensionality of the projected feature space. If we use the projected instances in the decision function, we obtain

$$f(\boldsymbol{x}) = \langle \boldsymbol{w},\Phi(\boldsymbol{z})\rangle + b$$

and the primal problem for SVM in (1) becomes

$$\begin{aligned} &\text{min.} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{w.r.t.} \quad \boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{W} \\ &\text{s.t.} \quad y_i(\langle \boldsymbol{w},\Phi(\boldsymbol{z}_i)\rangle + b) \geq 1-\xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i. \end{aligned} \tag{2}$$

Note that the optimization problem in (2) is not convex due to the nonlinearity in the separation constraints.

Instead of trying to optimize SVM parameters, $\{\boldsymbol{w},b,\boldsymbol{\xi}\}$, and the projection matrix, $\boldsymbol{W}$, together, we utilize a two-step optimization algorithm as in Chapelle et al. [14] and Rakotomamonjy et al. [17]. The algorithm starts with a random projection matrix. In the first step, we solve (2) with respect to $\{\boldsymbol{w},b,\boldsymbol{\xi}\}$ while fixing $\boldsymbol{W}$. We then update $\boldsymbol{W}$ using a gradient-descent step calculated from the objective function of (2) in the second step. The following dual formulation can be solved instead of the primal formulation in the first step to apply the kernel trick.

$$\begin{aligned} &\text{max.} \quad \mathrm{J} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \underbrace{\langle \Phi(\boldsymbol{z}_i),\Phi(\boldsymbol{z}_j)\rangle}_{K(\boldsymbol{z}_i,\boldsymbol{z}_j)} \\ &\text{w.r.t.} \quad \boldsymbol{\alpha} \\ &\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \quad \forall i. \end{aligned} \tag{3}$$

For a fixed $\boldsymbol{W}$, we solve the dual optimization problem and obtain the optimal $\boldsymbol{\alpha}$ values. We need to update $\boldsymbol{W}$ by calculating the gradient of the objective function in (3). The gradient of the objective function with respect to the elements of $\boldsymbol{W}$ is calculated as

$$\frac{\partial J}{\partial W_{kl}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \frac{\partial K(\boldsymbol{z}_i,\boldsymbol{z}_j)}{\partial W_{kl}}$$

where $k \in \{1,2,\ldots,d\}$ and $l \in \{1,2,\ldots,r\}$. The same gradient can also be obtained as the derivative of the margin [14].

Three commonly used kernels, linear kernel (K_L), polynomial kernel (K_P), and Gaussian kernel (K_G), can be expressed in terms of $\boldsymbol{W}$ as follows:

$$K_L(\boldsymbol{z}_i,\boldsymbol{z}_j) = \langle \boldsymbol{z}_i,\boldsymbol{z}_j\rangle = \boldsymbol{x}_i^\top \boldsymbol{W}\boldsymbol{W}^\top \boldsymbol{x}_j$$

$$K_P(\boldsymbol{z}_i,\boldsymbol{z}_j) = (\langle \boldsymbol{z}_i,\boldsymbol{z}_j\rangle + 1)^q = (\boldsymbol{x}_i^\top \boldsymbol{W}\boldsymbol{W}^\top \boldsymbol{x}_j + 1)^q$$

$$K_G(\boldsymbol{z}_i,\boldsymbol{z}_j) = \exp(-\|\boldsymbol{z}_i-\boldsymbol{z}_j\|^2/s^2) = \exp(-(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top \boldsymbol{W}\boldsymbol{W}^\top (\boldsymbol{x}_i-\boldsymbol{x}_j)/s^2).$$

The derivative of the kernels with respect to the elements of the projection matrix are given as

$$\frac{\partial K_L(\boldsymbol{z}_i,\boldsymbol{z}_j)}{\partial W_{kl}} = \boldsymbol{x}_i[k]\boldsymbol{z}_j[l] + \boldsymbol{z}_i[l]\boldsymbol{x}_j[k]$$

$$\frac{\partial K_P(\boldsymbol{z}_i,\boldsymbol{z}_j)}{\partial W_{kl}} = (\boldsymbol{x}_i[k]\boldsymbol{z}_j[l] + \boldsymbol{z}_i[l]\boldsymbol{x}_j[k])q(\langle \boldsymbol{z}_i,\boldsymbol{z}_j\rangle + 1)^{q-1}$$

$$\frac{\partial K_G(\boldsymbol{z}_i,\boldsymbol{z}_j)}{\partial W_{kl}} = -2(\boldsymbol{x}_i[k]-\boldsymbol{x}_j[k])(\boldsymbol{z}_i[l]-\boldsymbol{z}_j[l])K_G(\boldsymbol{z}_i,\boldsymbol{z}_j)/s^2$$

where $[\cdot]$ indexes the elements of a vector.

The projection matrix can be updated using a simple gradient-descent update rule with a fixed step size or Armijo's rule can be used to determine a better step size at each iteration (see Section 5). Note that this alternating optimization procedure does not guarantee convergence to the global optimum and the initial value of $\boldsymbol{W}$ may affect the solution quality. Algorithm 1 lists the main steps of the procedure which we call *global projection kernels* (GPK) from now on ($\Delta^{(t)}$ is the step size in gradient-descent).

Algorithm 1. Global projection kernels.

1: Initialize $\boldsymbol{W}$ to random numbers
2: **repeat**
3: Calculate $K(\boldsymbol{z}_i,\boldsymbol{z}_j)$
4: Solve canonical SVM with $K(\boldsymbol{z}_i,\boldsymbol{z}_j)$
5: $W_{kl}^{(t+1)} \Leftarrow W_{kl}^{(t)} - \Delta^{(t)}\frac{\partial J}{\partial W_{kl}} \quad \forall (k,l)$
6: **until** convergence

After convergence, we obtain the decision function in terms of model parameters as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\alpha_i y_i K(\boldsymbol{W}^\top \boldsymbol{x}_i, \boldsymbol{W}^\top \boldsymbol{x}) + b.$$

We project both the input $\boldsymbol{x}$ and the support vector $\boldsymbol{x}_i$ to the lower dimensional space and calculate the kernel there.

3. Localized kernel functions

Each mapping function we use in (1) or (2) corresponds to a different kernel function in the dual formulation and is directly related to the performance of the resulting classifier. The best kernel function (i.e., the mapping function) for a specific data set is generally selected from a set of candidate kernel functions using a statistical cross-validation procedure.

Instead of selecting and using a single mapping (kernel) function, Bach et al. [18] propose *multiple kernel learning* (MKL) which takes an unweighted sum of multiple discriminant values in different feature spaces obtained with different mapping functions:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}) \rangle + b$$

where m indexes kernels, $\boldsymbol{w}_m$ is the vector of weight coefficients, $\Phi_m(\cdot)$ is the mapping function for feature space m, and p is the number of kernels. The primal formulation of MKL is obtained as

$$\begin{aligned} \text{min.} \quad & \frac{1}{2}\left(\sum_{m=1}^{p} \|\boldsymbol{w}_m\|\right)^2 + C\sum_{i=1}^{n}\xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w}_m, b, \boldsymbol{\xi} \\ \text{s.t.} \quad & y_i\left(\sum_{m=1}^{p} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b\right) \geq 1-\xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i \end{aligned}$$

and the resulting decision function is a weighted sum of kernels:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{m=1}^{p} \alpha_i y_i \eta_m \underbrace{\langle \Phi_m(\boldsymbol{x}_i), \Phi_m(\boldsymbol{x}) \rangle}_{K_m(\boldsymbol{x}_i, \boldsymbol{x})} + b$$

where the kernel weights satisfy $\eta_m \geq 0$ and $\sum_{m=1}^{p} \eta_m = 1$.

MKL uses a fixed combination rule which assigns the same weight to a kernel over the whole input space. If data has an underlying local structure, we should give higher weights to appropriate kernel functions (i.e., kernels which match the complexity of data distribution) for each local region. Gönen and Alpaydın [15] propose *localized multiple kernel learning* (LMKL) by rewriting the discriminant function as follows, in order to allow local combinations of kernels:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \eta_m(\boldsymbol{x}|\boldsymbol{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}) \rangle + b \tag{4}$$

where $\eta_m(\boldsymbol{x}|\boldsymbol{V})$ is the *gating function* which chooses the weight for feature space m as a function of input $\boldsymbol{x}$ and $\boldsymbol{V}$ is the vector of the gating function parameters. Assuming that the regions of use of kernels are linearly separable, we can express the gating model as

$$\eta_m(\boldsymbol{x}|\boldsymbol{V}) = \frac{\exp(\langle \boldsymbol{v}_m, \Phi_{\mathcal{G}}(\boldsymbol{x}) \rangle + v_{m0})}{\sum_{k=1}^{p} \exp(\langle \boldsymbol{v}_k, \Phi_{\mathcal{G}}(\boldsymbol{x}) \rangle + v_{k0})} \tag{5}$$

where $\boldsymbol{V}$ includes all $\boldsymbol{v}_m$, v_{m0}, and $\Phi_{\mathcal{G}}(\boldsymbol{x})$ is the feature space in which we learn the gating model.

By modifying the original SVM formulation in (1) with the localized discriminant function in (4), we get the following optimization problem:

$$\begin{aligned} \text{min.} \quad & \frac{1}{2}\sum_{m=1}^{p} \|\boldsymbol{w}_m\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w}_m, b, \boldsymbol{\xi}, \boldsymbol{V} \\ \text{s.t.} \quad & y_i\left(\sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b\right) \geq 1-\xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i. \end{aligned}$$

LMKL uses an alternating optimization procedure to solve this nonconvex problem and obtains the discriminant function as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{m=1}^{p} \alpha_i y_i \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) K_m(\boldsymbol{x}_i, \boldsymbol{x}) \eta_m(\boldsymbol{x}|\boldsymbol{V}) + b.$$

4. Supervised learning of local projection kernels

Using a single projection matrix over the whole input space cannot capture multiple modalities that may exist in the data. At this point, we can combine localized kernel functions of Section 3 with projection matrices of Section 2 and similar to using kernel functions with changing weights in different regions, we can divide the input space into p regions and learn a local projection matrix, $\boldsymbol{W}_m \in \mathbb{R}^{d \times r}$, $m=1, \ldots, p$, in each region, in order to capture the local structure information. So, we have p different projected data instances for each instance:

$$\boldsymbol{z}_m = \boldsymbol{W}_m^{\top}\boldsymbol{x}, \quad m = 1, \ldots, p$$

and the discriminant function can be rewritten as

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \eta_m(\boldsymbol{x}|\boldsymbol{V}) \langle \boldsymbol{w}_m, \Phi(\boldsymbol{z}_m) \rangle + b$$

where the *gating function*, $\eta_m(\boldsymbol{x}|\boldsymbol{V})$, now chooses the weight for projected feature space m as a function of input $\boldsymbol{x}$. By modifying the formulation in (2) with this new discriminant function, we get the following optimization problem:

$$\begin{aligned} \text{min.} \quad & \frac{1}{2}\sum_{m=1}^{p} \|\boldsymbol{w}_m\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w}_m, b, \boldsymbol{\xi}, \boldsymbol{W}_m, \boldsymbol{V} \\ \text{s.t.} \quad & y_i\left(\sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) \langle \boldsymbol{w}_m, \Phi(\boldsymbol{z}_{im}) \rangle + b\right) \geq 1-\xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i \end{aligned} \tag{6}$$

and this problem is not convex, either. For given $\{\boldsymbol{W}_{1:p}, \boldsymbol{V}\}$ values, (6) becomes convex and we can obtain the Lagrangian of the primal problem:

$$\begin{aligned} L_D = & \frac{1}{2}\sum_{m=1}^{p} \|\boldsymbol{w}_m\|^2 + \sum_{i=1}^{n}(C-\alpha_i-\beta_i)\xi_i + \sum_{i=1}^{n}\alpha_i \\ & - \sum_{i=1}^{n}\alpha_i y_i\left(\sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) \langle \boldsymbol{w}_m, \Phi(\boldsymbol{z}_{im}) \rangle + b\right) \end{aligned}$$

and taking the derivatives of L_D with respect to the primal variables gives

$$\begin{aligned} & \frac{\partial L_D}{\partial \boldsymbol{w}_m} \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^{n} \alpha_i y_i \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) \Phi(\boldsymbol{z}_{im}) \quad \forall m \\ & \frac{\partial L_D}{\partial b} \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \frac{\partial L_D}{\partial \xi_i} \Rightarrow C = \alpha_i + \beta_i \quad \forall i. \end{aligned} \tag{7}$$

From (6) and (7), the dual formulation is obtained as

$$\text{max.} \quad \mathrm{J}_\eta = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

w.r.t. $\boldsymbol{\alpha}$

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \quad \forall i \tag{8}$$

where the *local projection kernel matrix* is defined as

$$K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) \underbrace{\langle \Phi(\boldsymbol{z}_{im}), \Phi(\boldsymbol{z}_{jm}) \rangle}_{K(\boldsymbol{W}_m^\top \boldsymbol{x}_i, \boldsymbol{W}_m^\top \boldsymbol{x}_j)} \eta_m(\boldsymbol{x}_j|\boldsymbol{V})$$

and using $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$ corresponds to projecting data instances into the $(p \times r)$- dimensional feature space and using the dot product in this feature space.

$$\begin{pmatrix} \eta_1(\boldsymbol{x}_i|\boldsymbol{V})\Phi(\boldsymbol{W}_1^\top \boldsymbol{x}_i) \\ \eta_2(\boldsymbol{x}_i|\boldsymbol{V})\Phi(\boldsymbol{W}_2^\top \boldsymbol{x}_i) \\ \vdots \\ \eta_p(\boldsymbol{x}_i|\boldsymbol{V})\Phi(\boldsymbol{W}_p^\top \boldsymbol{x}_i) \end{pmatrix}^\top \begin{pmatrix} \eta_1(\boldsymbol{x}_j|\boldsymbol{V})\Phi(\boldsymbol{W}_1^\top \boldsymbol{x}_j) \\ \eta_2(\boldsymbol{x}_j|\boldsymbol{V})\Phi(\boldsymbol{W}_2^\top \boldsymbol{x}_j) \\ \vdots \\ \eta_p(\boldsymbol{x}_j|\boldsymbol{V})\Phi(\boldsymbol{W}_p^\top \boldsymbol{x}_j) \end{pmatrix}$$

Having fixed SVM and gating, we can update the local projection matrices using gradient-descent. For given $\{\boldsymbol{\alpha},\boldsymbol{V}\}$ values, the gradient of the objective function in (8) with respect to the elements of $\boldsymbol{W}_m$ matrices is given as

$$\frac{\partial J_\eta}{\partial W_{mkl}} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \Upsilon_{ij} \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) \frac{\partial K(\boldsymbol{z}_{im},\boldsymbol{z}_{jm})}{\partial W_{mkl}} \eta_m(\boldsymbol{x}_j|\boldsymbol{V})$$

where $\Upsilon_{ij} = \alpha_i \alpha_j y_i y_j$.

Having fixed SVM and the local projections, we can update the gating parameters. For given $\{\boldsymbol{\alpha},\boldsymbol{W}_{1:p}\}$ values, the gradients of the objective function in (8) with respect to the gating model parameters are given as

$$\frac{\partial J_\eta}{\partial \boldsymbol{v}_m} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{p} \Upsilon_{ij} \eta_k(\boldsymbol{x}_i|\boldsymbol{V}) K(\boldsymbol{z}_{ik},\boldsymbol{z}_{jk}) \eta_k(\boldsymbol{x}_j|\boldsymbol{V}) \\ (\Phi_{\mathcal{G}}(\boldsymbol{x}_i)(\delta_m^k - \eta_m(\boldsymbol{x}_i|\boldsymbol{V})) + \Phi_{\mathcal{G}}(\boldsymbol{x}_j)(\delta_m^k - \eta_m(\boldsymbol{x}_j|\boldsymbol{V})))$$

$$\frac{\partial J_\eta}{\partial v_{m0}} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{p} \Upsilon_{ij} \eta_k(\boldsymbol{x}_i|\boldsymbol{V}) K(\boldsymbol{z}_{ik},\boldsymbol{z}_{jk}) \eta_k(\boldsymbol{x}_j|\boldsymbol{V}) \\ (\delta_m^k - \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) + \delta_m^k - \eta_m(\boldsymbol{x}_j|\boldsymbol{V}))$$

where δ_m^k is 1 if $m=k$, and 0 otherwise.

The complete algorithm of our proposed LPK with linear gating model is summarized in Algorithm 2 ($\Delta^{(t)}$ and $\mu^{(t)}$ are the step sizes of the corresponding updates in gradient-descent). The convergence of the algorithm can be determined by observing the change in the objective function value.

Algorithm 2. Local projection kernels.

1: Initialize $\{\boldsymbol{W}_{1:p},\boldsymbol{V}\}$ to random numbers
2: **repeat**
3: Calculate $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$
4: Solve canonical SVM with $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$
5: $W_{mkl}^{(t+1)} \Leftarrow W_{mkl}^{(t)} - \Delta^{(t)} \frac{\partial J_\eta}{\partial W_{mkl}} \quad \forall (m,k,l)$
6: Calculate $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$
7: Solve canonical SVM with $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$
8: $\boldsymbol{v}_m^{(t+1)} \Leftarrow \boldsymbol{v}_m^{(t)} - \mu^{(t)} \frac{\partial J_\eta}{\partial \boldsymbol{v}_m} \quad \forall m$
9: $v_{m0}^{(t+1)} \Leftarrow v_{m0}^{(t)} - \mu^{(t)} \frac{\partial J_\eta}{\partial v_{m0}} \quad \forall m$
10: **until** convergence

After determining the final $\{\boldsymbol{\alpha},b,\boldsymbol{W}_{1:p},\boldsymbol{V}\}$ values, the resulting discriminant function is

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \sum_{m=1}^{p} \alpha_i y_i \eta_m(\boldsymbol{x}_i|\boldsymbol{V}) K(\boldsymbol{W}_m^\top \boldsymbol{x}_i, \boldsymbol{W}_m^\top \boldsymbol{x}) \eta_m(\boldsymbol{x}|\boldsymbol{V}) + b.$$

In order to better illustrate the proposed method, we create a toy data set which consists of four clusters (two for each class) as shown in Fig. 1(a). If we use a global projection matrix over the whole input space, we cannot obtain a clear linear separation between classes due to intraclass multimodalities. However, we can obtain a projected space in which classes are well-separated and multimodal structures in each class are preserved, as shown in Fig. 1(b), by splitting the input space into two regions using the linear gating model (shown with the thick dashed line in Fig. 1(a)) and performing local projections (one-dimensional projections, $\boldsymbol{W}_1 \in \mathbb{R}^{2\times 1}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{2\times 1}$, shown with arrows in Fig. 1(a)) in each region.

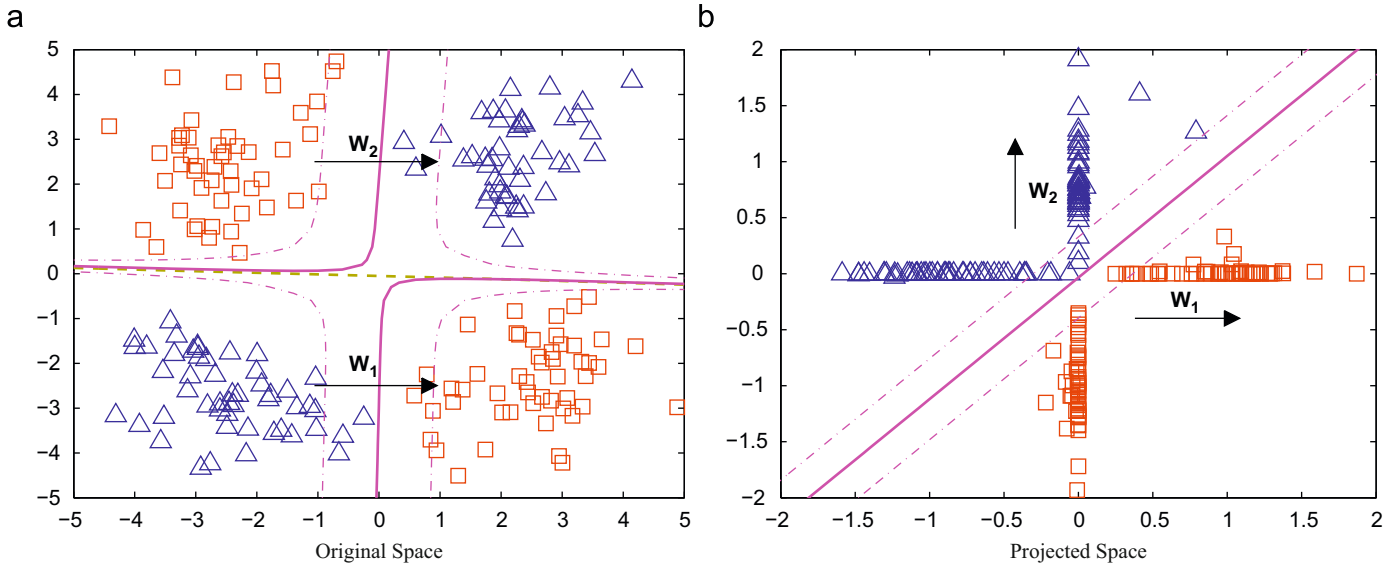

Fig. 1. Motivating example for learning local projections. (a) There are two local regions in the original feature space and the thick dashed line separate them. $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ arrows show the projection directions in the two regions. The solid lines show the discriminant in each region. (b) The horizontal and vertical axes correspond to the projected directions in the two regions and the solid line shows the resulting discriminant in this projected space. We see that the two classes are perfectly separated in this space.

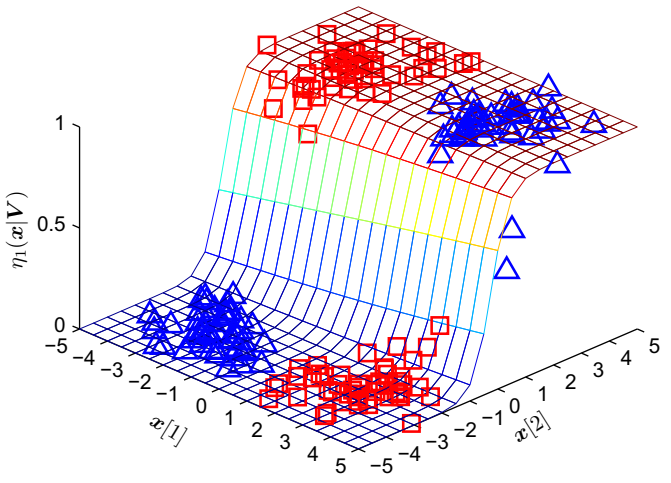

Fig. 2. The gating model output superimposed with training data for the motivating example.

Fig. 2 shows the gating model output superimposed with training data. We see that $\eta_1(\boldsymbol{x}|\boldsymbol{V})$ divides the input space into two local regions. The line where $\eta_1(\boldsymbol{x}|\boldsymbol{V})=0.5$ is shown by a thick dashed line in Fig. 1(a).

5. Discussion of the method

In LPK training, the gradient calculations have ignorable time complexity compared with the SVM solver and these calculations are made by using only the support vectors at the current iteration. The key issue for faster convergence is to select good gradient-descent step sizes ($\Delta^{(t)}$ and $\mu^{(t)}$ in Algorithm 2), at each iteration. Better step size values can be obtained by utilizing a line search method such as Armijo's rule but this process needs additional calls to the SVM solver. Clearly, the time complexity for each iteration increases but the algorithm converges in fewer iterations. In our experiments, we use Armijo's rule to determine the step sizes at each iteration and the algorithm converges in a few iterations (generally 5–10). A more detailed convergence analysis is performed in Section 6.4.

We describe LPK for binary classification problems but the same idea can easily be applied to regression estimation and novelty detection problems [1] by changing the dual optimization problem (8) solved at each iteration and calculating the gradients with respect to the new objective function. The gradient formulations obtained for binary classification problems can be used by just replacing Υ_{ij} with $(\alpha_i^+-\alpha_i^-)(\alpha_j^+-\alpha_j^-)$ for regression estimation and with $\alpha_i\alpha_j$ for novelty detection.

Using local projection matrices in different regions of the input space enables us to extract information about the relative importance of features in each region. The features with high magnitude weights in local projection matrices give more information in the corresponding region of the input space. The features with very small weights can also be discarded to perform feature selection locally.

Coupled learning of a data projection rule and a classification algorithm has also been studied by Weinberger et al. [19] and Globerson and Roweis [20]. In these studies, a Mahalanobis distance metric used in nearest neighbor classification is learned by directly considering the classification accuracy. Tao et al. [21] propose a supervised learning method that performs learning and feature extraction together for tensor data. The discriminant parameters and the projection matrix are optimized using an alternating approach. Our proposed LPK is more similar to Pereira and Gordon [22] in that the optimization of the projection matrix and the classifier (SVM as in our case) performed jointly. They use a global projection matrix over the whole input space, but we introduce a data-dependent projection by using a gating model for choosing the projection matrix.

6. Experiments

In this section, we evaluate the performance of the proposed method on visualization and classification tasks on benchmark data sets. We implement the main body of our algorithm in MATLAB and solve the optimization problems with MOSEK optimization software [23]. We stop the algorithm when the objective function value of the current iteration is not less than $(1-\tau)$ times the objective function value of the previous iteration. The parameter τ is set to 0.001 in our experiments (see Section 6.4). To compare, we use MATLAB implementation of LFDA [9] with default parameters.

6.1. Data visualization

We compare PCA, LFDA, and our proposed LPK for data visualization on small benchmark data sets, namely *Iris*, *Thyroid Disease*, *Letter Recognition*, and *Image Segmentation* from the UCI machine learning repository [24]. On these multiclass data sets, we merge certain classes, as done by Sugiyama et al. [9], to obtain multimodal two-class problems. In PCA and LFDA methods, we extract two dimensions by using the first two principal directions. In LPK method using SVM with the linear kernel, we use two regions ($p=2$) with the linear gating model and project data points to one dimension ($r=1$) in each region.

On *Iris*, we combine Setosa and Virginica into a single class to obtain multimodality. Fig. 3 shows the two-dimensional projected feature spaces found by each method. Both PCA and LFDA preserve within-class modality but could not achieve a clear between-class separation. However, our proposed LPK achieves a clear between-class separation while preserving within-class modality.

On *Thyroid Disease*, we merge Hypothyroidism and Hyperthyroidism classes into one class. As we can see from Fig. 4, all three methods obtain similar results but LPK has better separation between within-class modalities.

On *Letter Recognition*, we construct a two-class data set by combining 'A' and 'C' letters into one class versus 'B' letter in another class. LFDA achieves a good separation between clusters whereas PCA is not able to separate the samples from 'B' and 'C' letters (see Fig. 5). LPK also achieves good separation between different classes but it could not separate letters 'A' and 'C' as well as LFDA. This is mainly because of the discriminative nature of LPK, the main goal is to separate different classes rather than preserving multimodality in one class.

On *Image Segmentation*, we combine Brickface and Sky classes into one class and treat Foilage as another class. Fig. 6 shows that PCA and LFDA are not able to separate Brickface and Foilage classes, whereas LPK obtains three different clusters for each class while maintaining a good between-class separation.

6.2. Face recognition

We also compare PCA and LPK on the *Olivetti* face recognition data set in order to see the performance of LPK in a real-life scenario with a very high dimensional feature space. *Olivetti* data set consists of 10 different 64×64 grayscale images

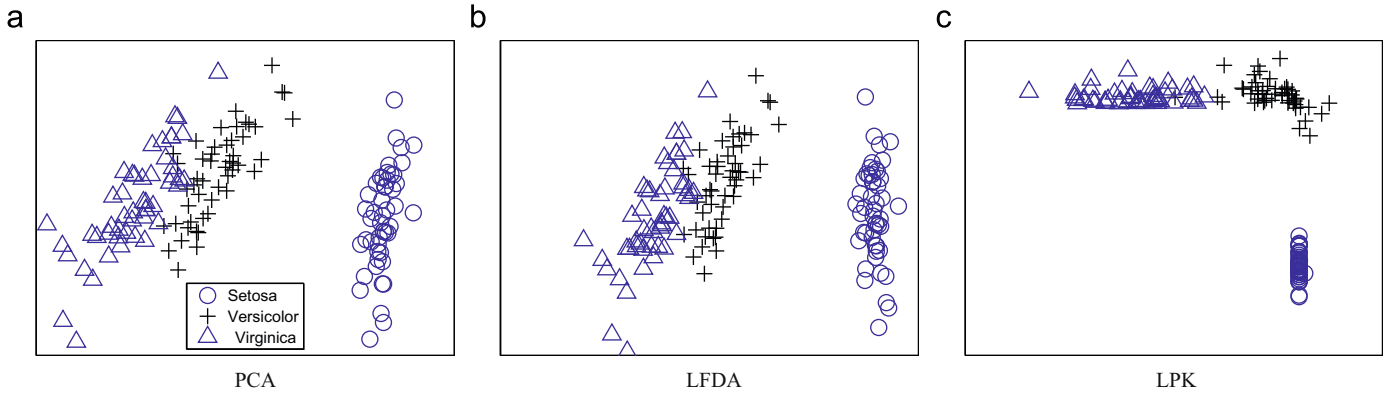

Fig. 3. Data visualization for *Iris* data set.

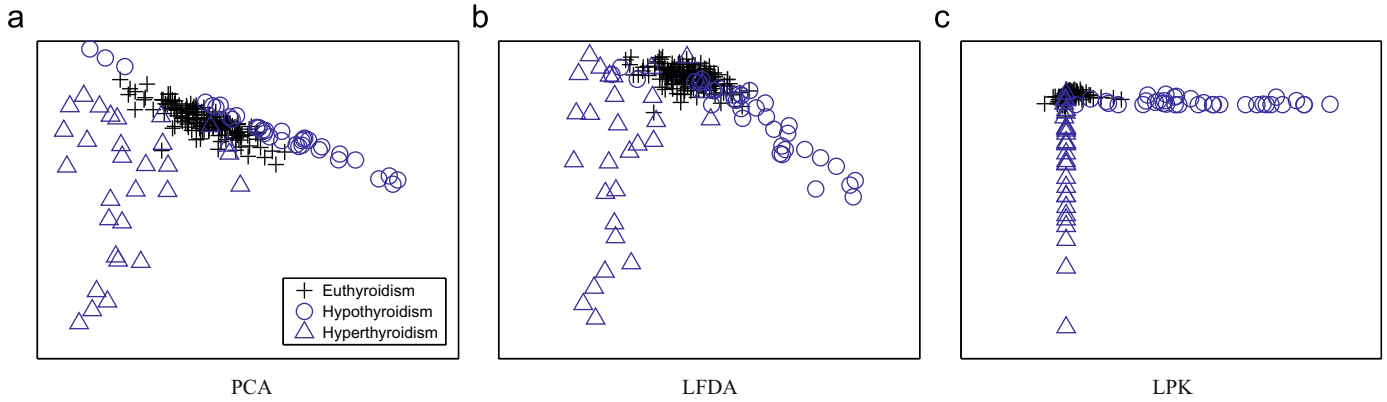

Fig. 4. Data visualization for *Thyroid Disease* data set.

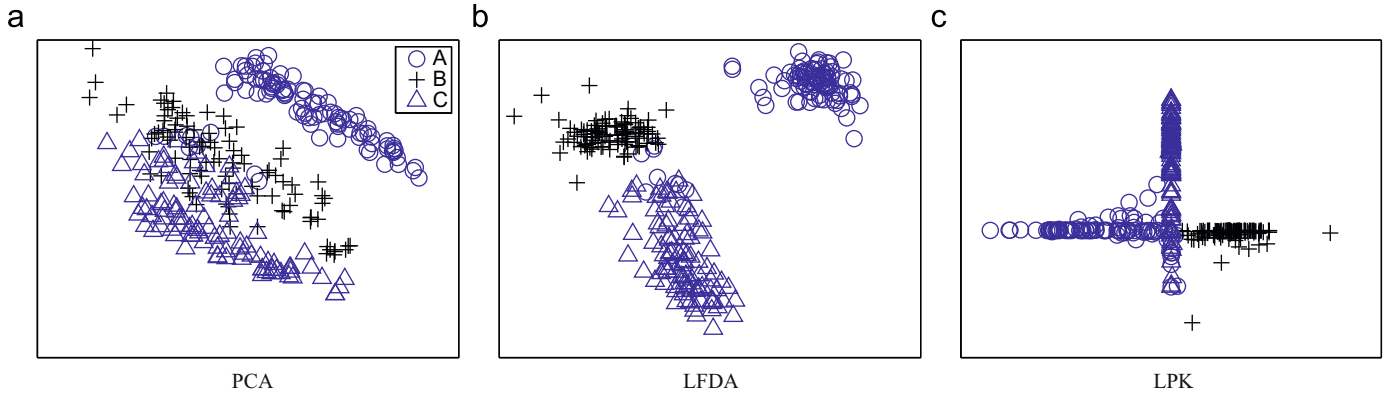

Fig. 5. Data visualization for *Letter Recognition* data set.

of 40 subjects. We construct a two-class data set by combining male subjects (36 subjects) into one class versus female subjects (four subjects) in another class. In both methods, we project data points to a two-dimensional space. PCA extracts these two dimensions by using the first two principal directions, and LPK using SVM with the linear kernel divides the original feature space into two regions ($p=2$) with the linear gating model and projects data points to one dimension ($r=1$) in each region.

Fig. 7(a) illustrates the projection obtained by PCA. We can see that PCA is not able to separate classes due to its unsupervised nature. Eigenfaces obtained from the first two principal eigenvectors are also shown on the two corners and they look like two male subjects.

LPK finds a better two-dimensional projected space as shown in Fig. 7(b). LPK is able to achieve a nearly perfect separation between classes except a single image. If we look at the face image produced from gating model parameters, $\{\boldsymbol{v}_1,\boldsymbol{v}_2\} \in \mathbb{R}^{4096\times 1}$, we can see that the gating model puts more emphasis on eyes, eyebrows, nose, and mouth to assign the weights to the local projection spaces for a given data instance. The face image obtained from the first local projection matrix, $\boldsymbol{W}_1 \in \mathbb{R}^{4096\times 1}$, is very much like a male subject with relatively higher weights on eyebrows and nose. The face image of the other local projection matrix, $\boldsymbol{W}_2 \in \mathbb{R}^{4096\times 1}$, looks like a female subject with relatively higher weights on eyes and mouth. LPK identifies the important parts of the face images without using any prior information while trying to optimize the separation between classes in a supervised manner.

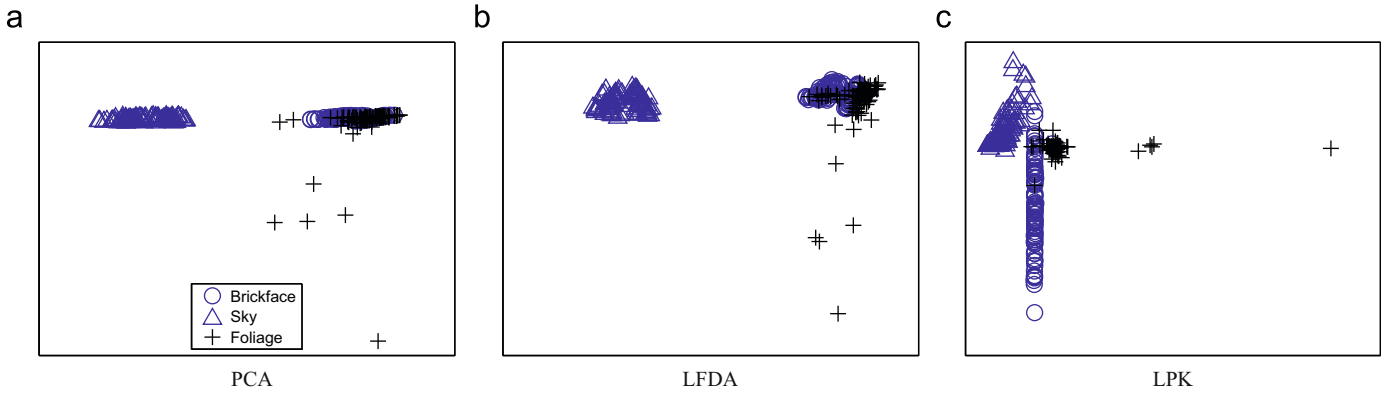

Fig. 6. Data visualization for *Image Segmentation* data set.

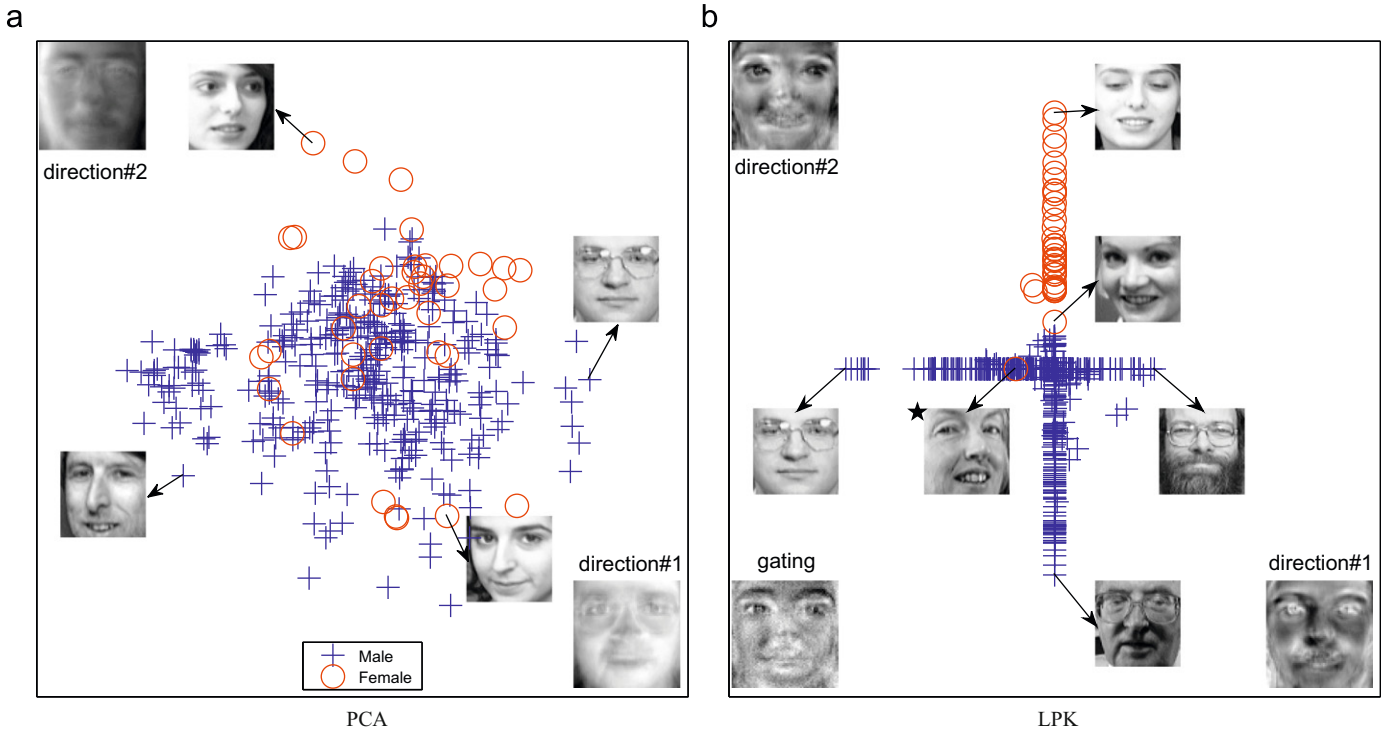

Fig. 7. Data visualization for *Olivetti* data set. (a) PCA: two eigenface images obtained from the first two principal eigenvectors are shown in the corners. (b) LPK ($p=2$ and $r=1$): the face images obtained from the local projection matrices, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, are shown in the corners. We also produce a face image from the gating model parameters, $\{\boldsymbol{v}_1,\boldsymbol{v}_2\}$, in order to see which features are important when dividing the input space into local regions. Only the image marked ★ is misclassified.

6.3. Classification accuracy

We evaluate the performance of PCA, LFDA, and LPK on classification tasks using large benchmark data sets. Table 1 lists the properties of the data sets. *Waveform* from the UCI repository is selected due to its multimodal structure and the first two classes are combined into a single one. *Usps-eo* (*Usps-sl*) is generated from *Usps* data set (16×16 grayscale digit images) by combining even (small: '0'–'4') numbers and odd (large: '5'–'9') numbers into different classes.

Our experimental methodology is as follows: Given a data set, a random one-third is reserved as the test set and the remaining two-thirds is resampled using 5×2 cross-validation to generate 10 training and validation sets, with stratification. Note that GPK algorithm discussed in Section 2 is equivalent to LPK with $p=1$. We also train SVMs after reducing dimensionality with PCA or LFDA using the same kernel in GPK and LPK (the linear kernel in our experiments). The validation sets of all folds are used to optimize C by trying values 0.01, 0.1, 1, 10, and 100. The best configuration (the one that has the highest average accuracy on the validation folds) is used to train the final SVMs on the training folds and their performance is measured over the test set. So, for each data set, we have 10 test set results; we display their averages and one standard deviation error bars.

Table 1
Classification data sets used in the experiments.

Data set	Dimensionality	# of instances
Waveform	21	1500
Usps-eo	256	1500
Usps-sl	256	1500

On *Waveform* (see Fig. 8), SVM trained after PCA and LFDA obtains nearly the same average accuracy results (around 89 per cent) after two dimensions. GPK achieves similar accuracy results with only one dimension. If we use local projection matrices with LPK ($p=2$ or 3), the average classification accuracy increases to 92 per cent using few dimensions. Because dimensionality reduction is done separately in different regions, we can work with much fewer dimensions attaining significantly higher accuracy. For example, when $r=2$ or 3, LPK ($p=2$ or 3) stores significantly fewer support vectors than SVM trained after PCA and LFDA while achieving significantly higher accuracy. Fitting a simpler model while attaining higher test accuracy is a clear indication of better generalization. SVM without any dimensionality reduction obtains 88.34 per cent average accuracy.

On *Usps-eo* (see Fig. 9), SVM trained after LFDA obtains an average accuracy around 79 per cent for all dimension values tried. However, SVM trained after PCA obtains better average accuracies after five dimensions and 86.10 per cent average accuracy with 15 dimensions. GPK and LPK ($p=3$) obtains more than 87 and 90 per cent average accuracy, respectively, for all dimension values tried. GPK and LPK achieve significantly higher accuracies and store significantly fewer support vectors than SVM trained after PCA for all configurations. SVM without any dimensionality reduction obtains 87.58 per cent average accuracy.

On *Usps-sl* (see Fig. 10), SVM trained after PCA has more than 70 per cent average accuracy after 14 dimensions whereas SVM trained after LFDA gets around 68 per cent average accuracy. GPK achieves average accuracy more than 75 per cent. When we use local projection matrices ($p=2$ or 3), the average accuracy increases to more than 86 per cent. LPK achieves 11 per cent higher accuracy than GPK and stores only 5 per cent of training instances as support vectors. SVM without any dimensionality reduction obtains 76.36 per cent average accuracy.

We also compare the classification performances of these methods on *Olivetti* (see Fig. 11). We use a different methodology for this data set. We select two images of each subject randomly and reserve these total 80 images as the test set. Then, we apply 8-fold cross-validation on the remaining 320 images by putting one image of each subject to the validation set at each fold. In order to get rid of singularity problems in LFDA method, we project data instances into a 100-dimensional space with PCA before applying LFDA. SVM trained after PCA could not achieve more than 96 per cent average accuracy whereas SVM trained after LFDA gets around 98 per cent average accuracy. GPK achieves average accuracy more than 98 per cent with four and five dimensions. LPK ($p=2$ or 3) achieves more than 98 per cent average accuracy after two dimensions ($r\geq 2$). For example, LPK ($p=2$ and $r=4$) obtains 99.69 per cent average accuracy. LPK stores nearly the same amount of support vectors as SVM trained after LFDA but achieves higher average accuracy. SVM without any dimensionality reduction obtains 99.06 per cent average accuracy.

6.4. Convergence analysis

We perform convergence analysis of LPK on *Waveform* and *Olivetti* data sets. We train LPK ($p=2$, $r=1$, and $C=100$) with the linear kernel for 25 iterations and record the objective function value, training and test set accuracies, and the percentage of support vectors at each iteration.

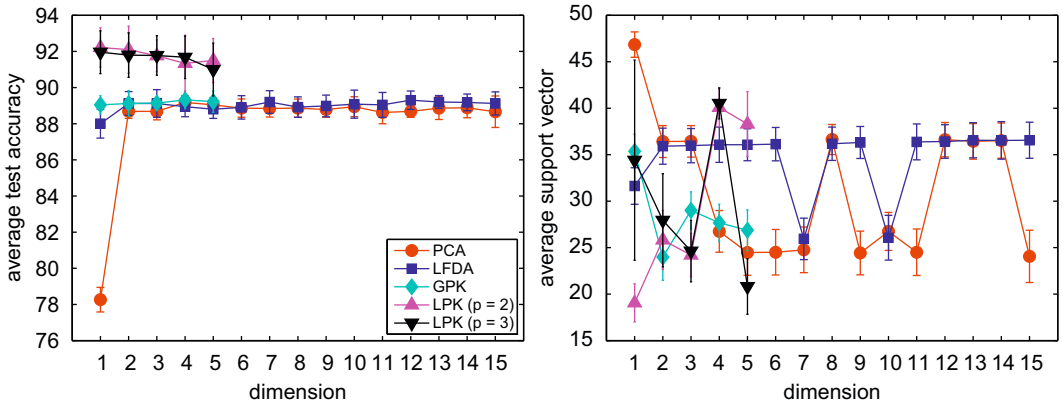

Fig. 8. Classification results for *Waveform* data set. The average test accuracy and the average percentage of support vectors versus the dimensionality of the projected space are plotted.

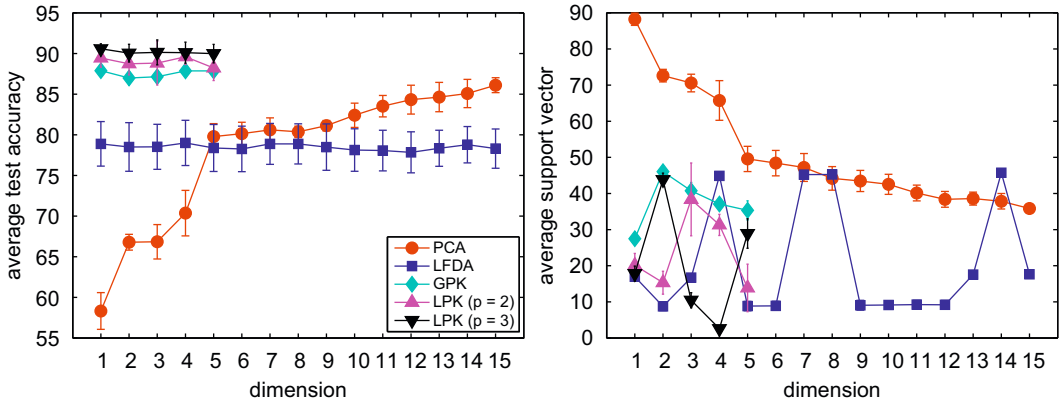

Fig. 9. Classification results for *Usps-eo* data set.

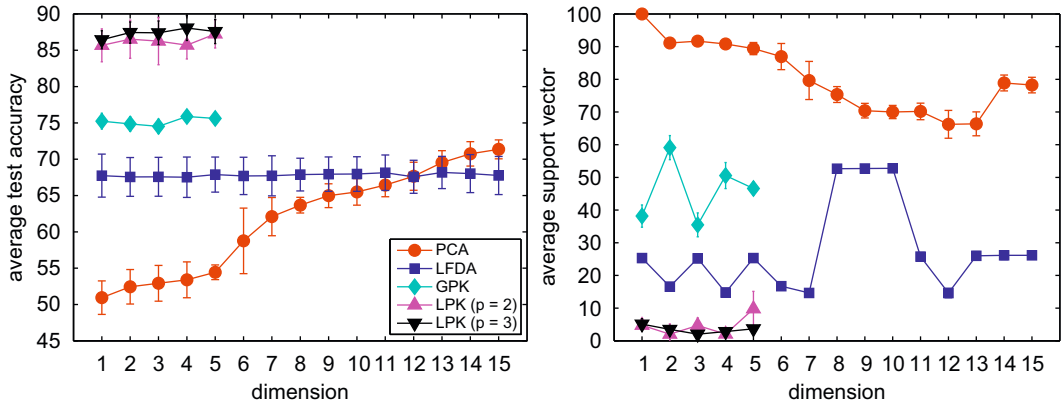

Fig. 10. Classification results for *Usps-sl* data set.

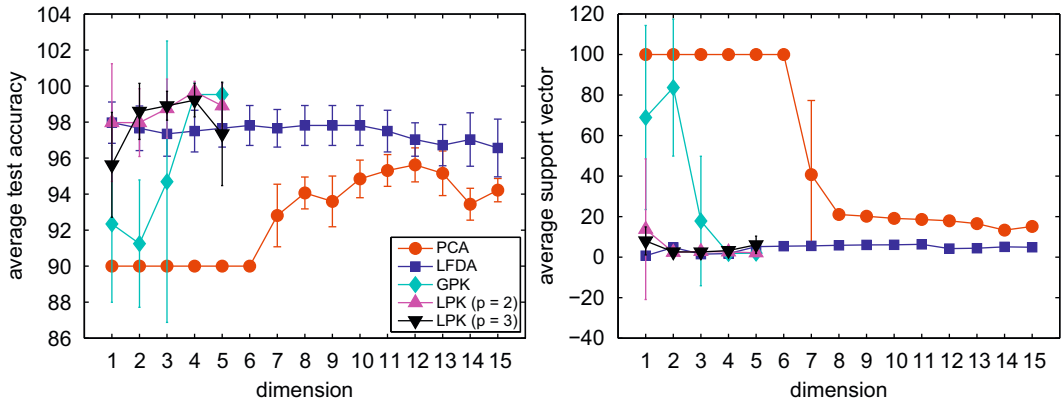

Fig. 11. Classification results for *Olivetti* data set.

Fig. 12 shows that LPK converges on *Waveform* data set after five iterations. If we use the stopping condition based on the objective function value with $\tau = 0.01$ or 0.001, LPK stops respectively after 9 and 13 iterations (shown with a square and a circle).

A similar behavior is also seen on *Olivetti* data set (see Fig. 13). We see that even $\tau = 0.01$ is too conservative. Note that on both data sets, LPK does not overfit even if we allow all 25 iterations.

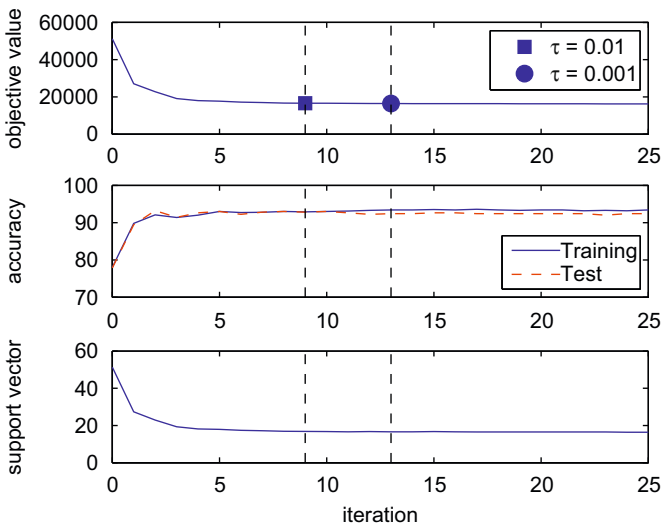

Fig. 12. Convergence analysis of LPK on *Waveform* data set.

7. Conclusions

In this work, we introduce a method for learning local projections coupled with a kernel-based learning algorithm. The proposed method has three main ingredients: (a) the gating model assigns weights to projection matrices for a data instance, (b) the local projection matrices perform dimensionality reduction separately in each region constructed by the gating model, (c) the kernel-based learning algorithm combines these locally constructed features.

The training of these three components are coupled, are all supervised, and the parameters of components are optimized together by using an alternating optimization scheme. The result of combining these three components is a *local projection kernel* which performs a locality preserving projection while considering the accuracy of the discriminant formed using such kernels. For binary classification tasks, the mathematical details of the proposed framework with linear gating are given. We discuss how the same derivation can be extended to regression estimation and novelty detection problems.

The proposed method, LPK, is tested and compared with two other algorithms, PCA and LFDA, for data visualization and classification tasks on benchmark data sets. On visualization tasks, LPK is able to maintain the multimodality of a class by placing clusters of the same class on the same side of the hyperplane while preserving a separation between them. This property is a direct result of using a gating model in LPK. On

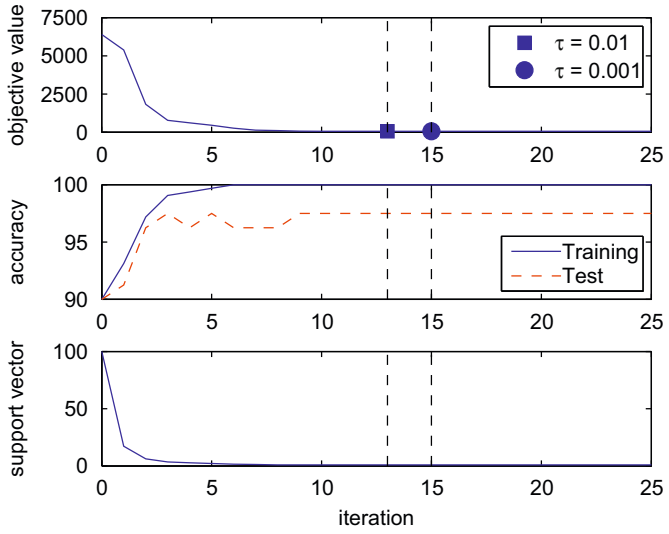

Fig. 13. Convergence analysis of LPK on *Olivetti* data set.

classification tasks, LPK achieves better results than PCA and LFDA by attaining both higher test accuracy and storing fewer support vectors due to the coupled optimization of the discriminant and the local projection matrices used in dimensionality reduction.

Acknowledgments

This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, Boğaziçi University Scientific Research Project 07HA101 and the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant EEEAG 107E222. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK.

References

[1] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, Cambridge, MA, 2002.
[2] M.E. Tipping, C.M. Bishop, Mixtures of probabilistic principal component analyzers, Neural Computation 11 (1999) 443–482.
[3] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[4] J. Tenenbaum, V. de Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[5] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Advances in Neural Information Processing Systems (NIPS), 2001.
[6] V. Ocnlinx, V. Wertz, M. Verleysen, Nonlinear data projection on non-Euclidean manifolds with controlled trade-off between trustworthiness and continuity, Neurocomputing 72 (2009) 1444–1454.
[7] C. Hou, J. Wang, Y. Wu, D. Yi, Local linear transformation embedding, Neurocomputing 72 (2009) 2368–2378.
[8] X. He, P. Niyogi, Locality preserving projections, in: Advances in Neural Information Processing Systems (NIPS), 2003.
[9] M. Sugiyama, Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis, Journal of Machine Learning Research 8 (2007) 1027–1061.
[10] D. Tao, X. Li, X. Wu, S.J. Maybank, Geometric mean for subspace selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 260–274.
[11] M. Wu, K. Yu, S. Yu, B. Schölkopf, Local learning projections, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 1039–1046.
[12] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 40–51.
[13] X. Li, S. Lin, S. Yan, D. Xu, Discriminant locally linear embedding with high-order tensor data, IEEE Transactions on Systems, Man, and Cybernetics, Part B (2008) 342–352.
[14] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (2002) 131–159.
[15] M. Gönen, E. Alpaydın, Localized multiple kernel learning, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 352–359.
[16] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[17] A. Rakotomamonjy, F.R. Bach, S. Canu, Y. Grandvalet, Simple MKL, Journal of Machine Learning Research 9 (2008) 2491–2521.
[18] F.R. Bach, G.R.G. Lanckriet, M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, in: ICML '04: Proceedings of the 21st International Conference on Machine Learning, 2004, p. 6.
[19] K.Q. Weinberger, J. Blitzer, L.K. Saul, Distance metric learning for large margin nearest neighbour classification, in: Advances in Neural Information Processing Systems (NIPS), 2005.
[20] A. Globerson, S. Roweis, Metric learning by collapsing classes, in: Advances in Neural Information Processing Systems (NIPS), 2005.
[21] D. Tao, X. Li, W. Hu, S. Maybank, X. Wu, Supervised tensor learning, in: Proceedings of the 5th IEEE International Conference on Data Mining, 2005.
[22] F. Pereira, G. Gordon, The support vector decomposition machine, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 689–696.
[23] Mosek, The MOSEK Optimization Tools Manual Version 5.0 (Revision 124), MOSEK ApS, Denmark, 2009.
[24] A. Asuncion, D. Newman, UCI machine learning repository, 2007 〈http://www.ics.uci.edu/∼mlearn/MLRepository.html〉.

Mehmet Gönen received the B.Sc. degree in industrial engineering and the M.Sc. degree in computer engineering from Boğaziçi University, İstanbul, Turkey, in 2003 and 2005, respectively, where he is currently working towards the Ph.D. degree at the Department of Computer Engineering. He is a Teaching Assistant at the Department of Computer Engineering, Boğaziçi University. His research interests include support vector machines, kernel methods, Bayesian methods, and real-time control and simulation of flexible manufacturing systems.

Ethem Alpaydın received his B.Sc. from the Department of Computer Engineering of Boğaziçi University in 1987 and the degree of Docteur es Sciences from Ecole Polytechnique Fédérale de Lausanne in 1990. He did his postdoctoral work at the International Computer Science Institute, Berkeley, in 1991 and afterwards was appointed as Assistant Professor at the Department of Computer Engineering of Boğaziçi University. He was promoted to Associate Professor in 1996 and Professor in 2002 in the same department. As visiting researcher, he worked at the Department of Brain and Cognitive Sciences of MIT in 1994, the International Computer Science Institute, Berkeley, in 1997 and IDIAP, Switzerland, in 1998. He was awarded a Fulbright Senior scholarship in 1997 and received the Research Excellence Award from the Boğaziçi University Foundation in 1998, the Young Scientist Award from the Turkish Academy of Sciences in 2001 and the Scientific Encouragement Award from the Scientific and Technological Research Council of Turkey in 2002. His book Introduction to Machine Learning was published by The MIT Press in October 2004. Its German edition was published in 2008, its Chinese edition in 2009, and its second edition in 2010. Its Turkish edition is in preparation. He is a senior member of the IEEE, an editorial board member of The Computer Journal (Oxford University Press) and an associate editor of Pattern Recognition (Elsevier).