Localized Multiple Kernel Learning

Mehmet Gönen GONEN@BOUN.EDU.TR
Ethem Alpaydın ALPAYDIN@BOUN.EDU.TR
Department of Computer Engineering, Boğaziçi University, TR-34342, Bebek, İstanbul, Turkey

Abstract

Recently, instead of selecting a single kernel, multiple kernel learning (MKL) has been proposed which uses a convex combination of kernels, where the weight of each kernel is optimized during training. However, MKL assigns the same weight to a kernel over the whole input space. In this paper, we develop a localized multiple kernel learning (LMKL) algorithm using a gating model for selecting the appropriate kernel function locally. The localizing gating model and the kernel-based classifier are coupled and their optimization is done in a joint manner. Empirical results on ten benchmark and two bioinformatics data sets validate the applicability of our approach. LMKL achieves statistically similar accuracy results compared with MKL by storing fewer support vectors. LMKL can also combine multiple copies of the same kernel function localized in different parts. For example, LMKL with multiple linear kernels gives better accuracy results than using a single linear kernel on bioinformatics data sets.

1. Introduction

Kernel-based methods such as the support vector machine (SVM) gained much popularity due to their success. For classification tasks, the basic idea is to map the training instances from the input space to a feature space (generally a higher dimensional space than the input space) where they are linearly separable. The SVM discriminant function obtained after training is:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b \tag{1}$$

where $\boldsymbol{w}$ is the weight coefficients, b is the threshold, and $\Phi(\boldsymbol{x})$ is the mapping function to the corresponding feature space. We do not need to define the mapping function explicitly and if we plug $\boldsymbol{w}$ vector from dual formulation into (1), we obtain the discriminant:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \underbrace{\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_i) \rangle}_{K(\boldsymbol{x}, \boldsymbol{x}_i)} + b$$

where n is the number of training instances, $\boldsymbol{x}_i$, and $K(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_i) \rangle$ is the corresponding kernel.

Each $\Phi(\boldsymbol{x})$ function has its own characteristics and corresponds to a different kernel function and leads to a different discriminant function in the original space. Selecting the kernel function (i.e., selecting the mapping function) is an important step in SVM training and is generally performed using cross-validation.

In recent studies (Lanckriet et al., 2004a; Sonnenburg et al., 2006), it is reported that using multiple different kernels instead of a single kernel improves the classification performance. The simplest way is to use an unweighted sum of kernel functions (Pavlidis et al., 2001; Moguerza et al., 2004). Using an unweighted sum gives equal preference to all kernels and this may not be ideal. A better strategy is to learn a weighted sum (e.g., convex combination); this also allows extracting information from the weights assigned to kernels. Lanckriet et al. (2004b) formulate this as a semidefinite programming problem which allows finding the combination weights and support vector coefficients together. Bach et al. (2004) reformulate the problem and propose an efficient algorithm using sequential minimal optimization (SMO). Their discriminant function can be seen as an unweighted summation of discriminant values (but a weighted summation of kernel functions) in different feature spaces:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}) \rangle + b \tag{2}$$

where m indexes kernels, $\boldsymbol{w}_m$ is the weight coefficients, $\Phi_m(\boldsymbol{x})$ is the mapping function for feature space m, and p is the number of kernels. By plugging $\boldsymbol{w}_m$ de-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

rived from duality conditions into (2), we obtain:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \eta_m \sum_{i=1}^{n} \alpha_i y_i \underbrace{\langle \Phi_m(\boldsymbol{x}), \Phi_m(\boldsymbol{x}_i) \rangle}_{K_m(\boldsymbol{x}, \boldsymbol{x}_i)} + b \qquad (3)$$

where the kernel weights satisfy $\eta_m \geq 0$ and $\sum_{m=1}^{p} \eta_m = 1$. The kernels we combine can be the same kernel with different hyperparameters (e.g., degree in polynomial kernel) or different kernels (e.g., linear, polynomial, and Gaussian kernels). We can also combine kernels over different data representations or different feature subsets.

Using a fixed combination rule (unweighted or weighted) assigns the same weight to a kernel over the whole input space. Assigning different weights to a kernel in different regions of the input space may produce a better classifier. If data has underlying localities, we should give higher weights to appropriate kernel functions (i.e., kernels which match the complexity of data distribution) for each local region. Lewis et al. (2006) propose to use a nonstationary combination method derived with a large-margin latent variable generative method. They use a log-ratio of Gaussian mixtures as the classifier. Lee et al. (2007) combine Gaussian kernels with different width parameters to capture the underlying local distributions, by forming a compositional kernel matrix from Gaussian kernels and using it to train a single classifier.

In this paper, we introduce a localized formulation of the multiple kernel learning (MKL) problem. In Section 2, we modify the discriminant function of the MKL framework proposed by Bach et al. (2004) with a localized one and describe how to optimize the parameters with a two-step optimization procedure. Section 3 explains the key properties of the proposed algorithm. We then demonstrate the performance of our localized multiple kernel learning (LMKL) method on toy, benchmark, and bioinformatics data sets in Section 4. We conclude in Section 5.

2. Localized Multiple Kernel Learning

We describe the LMKL framework for binary classification SVM but the derivations in this section can easily be extended to other kernel-based learning algorithms. We propose to rewrite the discriminant function (2) of Bach et al. (2004) as follows, in order to allow local combinations of kernels:

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

where $\eta_m(\boldsymbol{x})$ is the *gating function* which chooses feature space m as a function of input $\boldsymbol{x}$. $\eta_m(\boldsymbol{x})$ is defined up to a set of parameters which are also learned from data, as we will discuss below. By modifying the original SVM formulation with this new discriminant function, we get the following optimization problem:

$$\begin{aligned} &\min \frac{1}{2} \sum_{m=1}^{p} \|\boldsymbol{w}_m\|^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{w.r.t. } \boldsymbol{w}_m, b, \boldsymbol{\xi}, \eta_m(\boldsymbol{x}) \\ &\quad \text{s.t. } y_i \left(\sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \quad \forall i \\ &\qquad \xi_i \geq 0 \quad \forall i \end{aligned} \qquad (5)$$

where C is the regularization parameter and $\boldsymbol{\xi}$ is the slack variables as usual. Note that the optimization problem in (5) is not convex due to the nonlinearity introduced in the separation constraints.

Instead of trying to solve (5) directly, we can use a two-step alternate optimization algorithm inspired from Rakotomamonjy et al. (2007), to find the parameters of $\eta_m(\boldsymbol{x})$ and the discriminant function. The first step is to solve (5) with respect to $\boldsymbol{w}_m$, b, and $\boldsymbol{\xi}$ while fixing $\eta_m(\boldsymbol{x})$ and the second step is to update the parameters of $\eta_m(\boldsymbol{x})$ using a gradient-descent step calculated from the objective function in (5). The objective value obtained for a fixed $\eta_m(\boldsymbol{x})$ is an upper bound for (5) and the parameters of $\eta_m(\boldsymbol{x})$ are updated according to the current solution. The objective value obtained at the next iteration can not be greater than the current one due to the use of gradient-descent procedure and as iterations progress with a proper step size selection procedure (see Section 3.1), the objective value of (5) never increases. Note that this does not guarantee convergence to the global optimum and the initial parameters of $\eta_m(\boldsymbol{x})$ may affect the solution quality.

For a fixed $\eta_m(\boldsymbol{x})$, we obtain the Lagrangian of the primal problem in (5) as follows:

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

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

$$\begin{aligned} &\frac{\partial L_D}{\partial \boldsymbol{w}_m} \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^{n} \alpha_i y_i \eta_m(\boldsymbol{x}_i) \Phi_m(\boldsymbol{x}_i) \quad \forall m \\ &\frac{\partial L_D}{\partial b} \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\frac{\partial L_D}{\partial \xi_i} \Rightarrow C = \alpha_i + \beta_i \quad \forall i \, . \end{aligned} \qquad (6)$$

From (5) and (6), the dual formulation is obtained as:

$$\begin{aligned}
\max\ & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{w.r.t. } & \boldsymbol{\alpha} \\
\text{s.t. } & \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i
\end{aligned} \tag{7}$$

where the *locally combined kernel matrix* is defined as:

$$K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i) \underbrace{\langle \Phi_m(\boldsymbol{x}_i), \Phi_m(\boldsymbol{x}_j) \rangle}_{K_m(\boldsymbol{x}_i, \boldsymbol{x}_j)} \eta_m(\boldsymbol{x}_j) \ .$$

This formulation corresponds to solving a canonical SVM dual problem with the kernel matrix $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which should be positive semidefinite. We know that multiplying a kernel function with outputs of a nonnegative function for both input instances, known as quasi-conformal transformation, gives a positive semidefinite kernel matrix (Amari & Wu, 1998). So, the locally combined kernel matrix can be viewed as applying a quasi-conformal transformation to each kernel function and summing them to construct a combined kernel matrix. The only restriction is to have nonnegative $\eta_m(\boldsymbol{x})$ to get a positive semidefinite kernel matrix.

Choosing among possible kernels can be considered as a classification problem and we assume that the regions of use of kernels are linearly separable. In this case, the gating model can be expressed as:

$$\eta_m(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{v}_m, \boldsymbol{x} \rangle + v_{m0})}{\sum\limits_{k=1}^{p} \exp(\langle \boldsymbol{v}_k, \boldsymbol{x} \rangle + v_{k0})}$$

where $\boldsymbol{v}_m, v_{m0}$ are the parameters of this gating model and the softmax guarantees nonnegativity. One can use more complex gating models for $\eta_m(\boldsymbol{x})$ or equivalently implement the gating not in the original input space but in a space defined by a basis function, which can be one or some combination of the $\Phi_m(\boldsymbol{x})$ in which the SVM works (thereby also allowing the use of nonvectorial data). If we use a gating model which is constant (not a function of $\boldsymbol{x}$), our algorithm finds a fixed combination over the whole input space, similar to the original MKL formulation.

The proposed method differs from taking subsets of the training set and training a classifier in each subset then combining them. For example, Collobert et al. (2001) define such a procedure which learns an independent SVM for each subset and reassigns instances to subsets by training a gating model with a cost function. Our approach is different in that LMKL couples subset selection and combination of local classifiers in a joint optimization problem. LMKL is similar to but also different from the mixture of experts framework (Jacobs et al., 1991) in the sense that the gating model combines kernel-based experts and is learned together with experts; the difference is that in the mixture of experts, experts individually are classifiers whereas in our formulation, there is no discriminant per kernel.

For a given $\eta_m(\boldsymbol{x})$, we can say that the objective value of (7) is equal to the objective value of (5) due to strong duality. We can safely use the objective function of (7) as $J(\eta)$ function to calculate the gradients of the primal objective with respect to the parameters of $\eta_m(\boldsymbol{x})$. To train the gating model, we take derivatives of $J(\eta)$ with respect to $\boldsymbol{v}_m, v_{m0}$ and use gradient-descent:

$$\begin{aligned}
\frac{\partial J(\eta)}{\partial v_{m0}} &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{p} \alpha_i \alpha_j y_i y_j \eta_k(\boldsymbol{x}_i) K_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\qquad \eta_k(\boldsymbol{x}_j) \Big(\delta_m^k - \eta_m(\boldsymbol{x}_i) + \delta_m^k - \eta_m(\boldsymbol{x}_j) \Big) \\
\frac{\partial J(\eta)}{\partial \boldsymbol{v}_m} &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{p} \alpha_i \alpha_j y_i y_j \eta_k(\boldsymbol{x}_i) K_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\qquad \eta_k(\boldsymbol{x}_j) \Big(\boldsymbol{x}_i \left[\delta_m^k - \eta_m(\boldsymbol{x}_i) \right] + \boldsymbol{x}_j \left[\delta_m^k - \eta_m(\boldsymbol{x}_j) \right] \Big)
\end{aligned}$$

where δ_m^k is 1 if $m = k$ and 0 otherwise. After updating the parameters of $\eta_m(\boldsymbol{x})$, we are required to solve a single kernel SVM with $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ at each step.

The complete algorithm of LMKL with the linear gating model is summarized in Algorithm 1. Convergence of the algorithm can be determined by observing the change in $\boldsymbol{\alpha}$ or the parameters of $\eta_m(\boldsymbol{x})$.

Algorithm 1 LMKL with the linear gating model

```
1: Initialize $\boldsymbol{v}_m$ and $v_{m0}$ to small random numbers
   for $m = 1, \dots, p$
2: repeat
3:     Calculate $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with gating model
4:     Solve canonical SVM with $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$
5:     $v_{m0}^{(t+1)} \Leftarrow v_{m0}^{(t)} - \mu^{(t)} \frac{\partial J(\eta)}{\partial v_{m0}}$ for $m = 1, \dots, p$
6:     $\boldsymbol{v}_m^{(t+1)} \Leftarrow \boldsymbol{v}_m^{(t)} - \mu^{(t)} \frac{\partial J(\eta)}{\partial \boldsymbol{v}_m}$ for $m = 1, \dots, p$
7: until convergence
```

After determining the final $\eta_m(\boldsymbol{x})$ and SVM solution, the resulting discriminant function is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \sum_{m=1}^{p} \alpha_i y_i \eta_m(\boldsymbol{x}) K_m(\boldsymbol{x}, \boldsymbol{x}_i) \eta_m(\boldsymbol{x}_i) + b \ . \tag{8}$$

3. Discussions

We explain the key properties and possible extensions of the proposed algorithm in this section.

3.1. Computational Complexity

In each iteration, we are required to solve a canonical SVM problem with the combined kernel obtained with the current gating model and to calculate the gradients of $J(\eta)$. The gradient calculation step has ignorable time complexity compared to the SVM solver. The step size of each iteration, $\mu^{(t)}$, should be determined with a line search method which requires additional SVM optimizations for better convergence. The computational complexity of our algorithm mainly depends on the complexity of the canonical SVM solver used in the main loop, which can be reduced by using hot-start (i.e., giving previous $\boldsymbol{\alpha}$ as input). The number of iterations before convergence clearly depends on the training data and the step size selection procedure. The time complexity for testing is also reduced as a result of localizing. $K_m(\boldsymbol{x}, \boldsymbol{x}_i)$ in (8) needs to be evaluated only if both $\eta_m(\boldsymbol{x})$ and $\eta_m(\boldsymbol{x}_i)$ are nonzero.

3.2. Extensions to Other Kernel-Based Algorithms

LMKL can also be applied to kernel-based algorithms other than binary classification SVM, such as regression and one-class SVMs. We need to make two basic changes: (a) optimization problem and (b) gradient calculations from the objective value found. Otherwise, the same algorithm applies.

3.3. Knowledge Extraction

The MKL framework is used to extract knowledge about the relative contributions of kernel functions used in combination. If kernel functions are evaluated over different feature subsets or data representations, the important ones have higher combination weights. With our LMKL framework, we can deduce similar information based on different regions of the input space.

Our proposed method also allows combining multiple copies of the same kernel to obtain localized discriminants, thanks to the nonlinearity introduced by the gating model. For example, we can combine linear kernels with the gating model to obtain nearly piecewise linear boundaries.

4. Experiments

We implement the main body of our algorithm in C++ and solve the optimization problems with MOSEK optimization software (Mosek, 2008). 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 ten training and validation sets, with stratification. 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 ten test set results.

We perform simulations with three commonly used kernels: linear kernel (K_L), polynomial kernel (K_P), and Gaussian kernel (K_G):

$$\begin{aligned} K_L(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ K_P(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^q \\ K_G(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 / s^2\right) . \end{aligned}$$

We use the second degree ($q = 2$) polynomial kernel and estimate s in the Gaussian kernel as the average nearest neighbor distance between instances of the training set. All kernel matrices are calculated and normalized to unit trace before training. The step size of each iteration, $\mu^{(t)}$, is fixed as 0.01 without performing line search and a total of 50 iterations are performed.

4.1. Toy Data Set

In order to illustrate our proposed algorithm, we create a toy data set, named GAUSS4, which consists of 1200 data instances generated from four Gaussian components (two for each class) with the following prior probabilities, mean vectors and covariance matrices:

$$\begin{aligned} p_{11} &= 0.25 \quad \mu_{11} = \begin{pmatrix} -3.0 \\ +1.0 \end{pmatrix} \quad \Sigma_{11} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix} \\ p_{12} &= 0.25 \quad \mu_{12} = \begin{pmatrix} +1.0 \\ +1.0 \end{pmatrix} \quad \Sigma_{12} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix} \\ p_{21} &= 0.25 \quad \mu_{21} = \begin{pmatrix} -1.0 \\ -2.2 \end{pmatrix} \quad \Sigma_{21} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix} \\ p_{22} &= 0.25 \quad \mu_{22} = \begin{pmatrix} +3.0 \\ -2.2 \end{pmatrix} \quad \Sigma_{22} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix} \end{aligned}$$

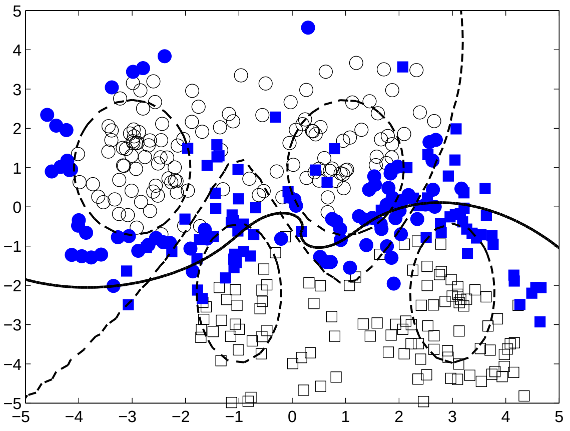
where data instances from the first two components are of class 1 (labeled as positive) and others are of class 2 (labeled as negative)[1]. We perform two sets of experiments on GAUSS4 data set: (K_L-K_P) and (K_L-K_L-K_L).

[1]MATLAB implementation of LMKL with an SMO-based canonical SVM solver and GAUSS4 dataset are available at `http://www.cmpe.boun.edu.tr/~gonen/lmkl`.

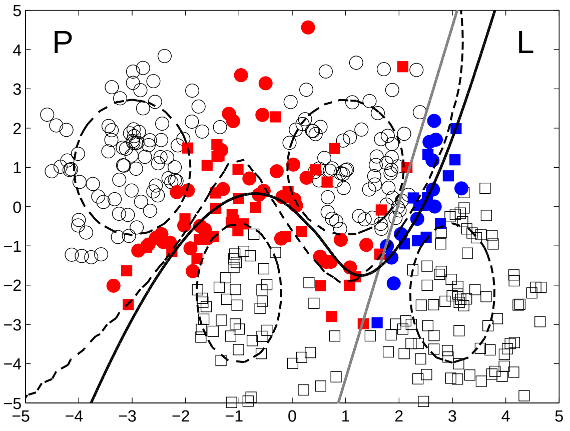
First, we train both MKL and LMKL for (K_L-K_P) combination. Figure 1(a) shows the classification boundaries calculated and the support vectors stored by MKL which assigns combination weights 0.30 and 0.70 to K_L and K_P, respectively. Using the kernel matrix obtained combining K_L and K_P with these weights, we do not achieve a good approximation to the optimal Bayes' boundary. As we see in Figure 1(b), LMKL divides the input space into two regions and uses the polynomial kernel to separate one component from two others quadratically and the linear kernel for the other component. We see that the locally combined kernel matrix obtained from K_L and K_P with the linear gating model learns a classification boundary very similar to the optimal Bayes' boundary. Note that the softmax function in the gating model achieves a smooth transition between kernels.

The effect of combining multiple copies of the same kernel can be seen in Figure 1(c) which shows the classification and gating model boundaries of LMKL with (K_L-K_L-K_L) combination. Using linear kernels in three different regions enables us to approximate the optimal Bayes' boundary in a piecewise linear manner. Instead of using complex kernels such as the Gaussian kernel, local combination of simple kernels (e.g., linear and polynomial kernels) can produce accurate classifiers and avoid overfitting. For example, the Gaussian kernel achieves 89.67 per cent average testing accuracy by storing all training instances as support vectors. However, LMKL with three linear kernels achieves 92.00 per cent average testing accuracy by storing 23.18 per cent of training instances as support vectors on the average.

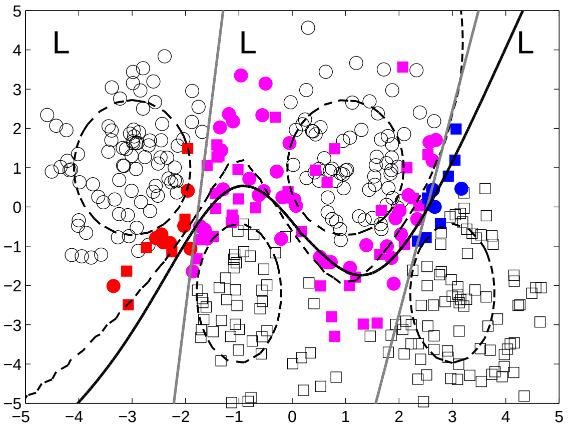
Initially, we assign small random numbers to the gating model parameters and this gives nearly equal combination weights for each kernel. This is equivalent to taking an unweighted summation of the original kernel matrices. The gating model starts to give crisp outputs as iterations progress and the locally combined kernel matrix becomes more sparse (see Figure 2). The kernel function values between data instances from different regions become 0 due to the multiplication of the gating model outputs. This localizing characteristics is also effective for the test instances. If the gating model gives crisp outputs for a test instance, the discriminant function in (8) is calculated over only the support vectors having nonzero gating model outputs for the selected kernels. Hence, discriminant function value for a data instance is mainly determined by the neighboring training instances and the active kernel function in its region.



(a) MKL with (K_L-K_P).



(b) LMKL with (K_L-K_P).



(c) LMKL with (K_L-K_L-K_L).

Figure 1. Separating hyperplanes (black solid lines) and support vectors (filled points) on GAUSS4 data set. Dashed lines show the Gaussians from which data are sampled and the optimal Bayes' discriminant. The gray solid lines shows the boundaries calculated from the gating models by considering them as classifiers which select a kernel function.

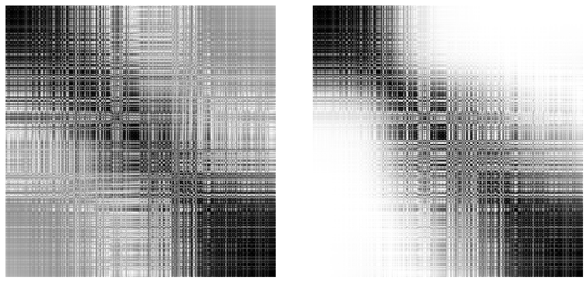

(a) First iteration (b) Last iteration

Figure 2. Locally combined kernel matrices, $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, of LMKL with ($K_L$-$K_P$) on Gauss4 data set.

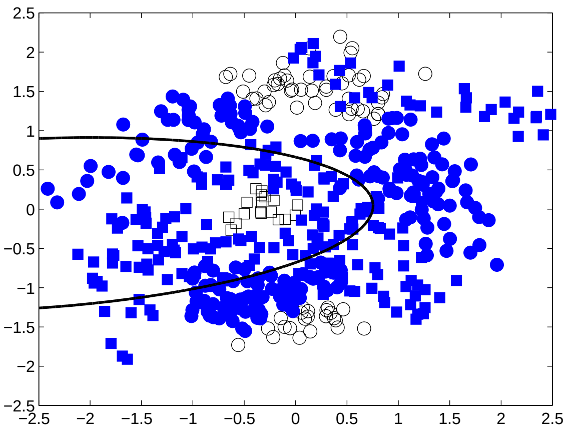
4.2. Benchmark Data Sets

We perform experiments on ten two-class benchmark data sets from the UCI machine learning repository and Statlog collection. In the result tables, we report the average testing accuracies and support vector percentages. The average accuracies and support vector percentages are made bold if the difference between the two compared classifiers is significant using the 5×2 cross-validation paired F test (Alpaydın, 1999).

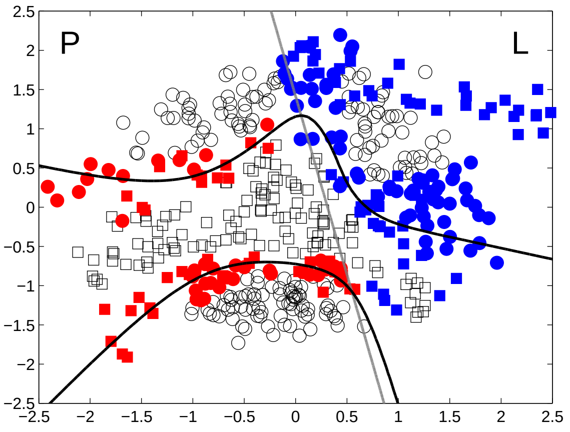
Figure 3(a)-(b) illustrate the difference between MKL and LMKL on Banana data set with (K_L-K_P) combination. We can see that MKL can not capture the localities exist in the data by combining linear and polynomial kernels with fixed combination weights (it assigns 1.00 to K_P ignoring the linear kernel). However, LMKL finds a more reasonable decision boundary using much fewer support vectors by dividing the input space into two regions using the linear gating model. The average testing accuracy increases from 70.52 to 84.46 per cent and the support vector count is halved (decreases from 82.36 to 41.28 per cent).

The classification and gating model boundaries found by LMKL with (K_L-K_L-K_L) combination on Banana data set can be seen in Figure 3(c). The gating model divides the input space into three regions and in each region a local and (nearly) linear decision boundary is induced. Combination of these local boundaries with softmax gating gives us a more complex boundary.

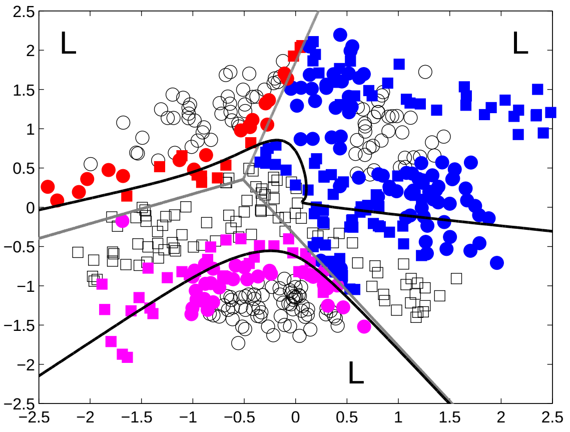
The results by MKL and LMKL for (K_P-K_G) and canonical SVMs with K_L, K_P, K_G are given in Table 1. LMKL achieves statistically similar accuracies compared with MKL on all data sets. LMKL stores significantly fewer support vectors on Heart, Pima, and Wdbc data sets. With direct comparison of average values, the localized variant performs better on seven and eight out of ten data sets in terms of testing accuracy and support vector percentage, respectively. Other kernel combinations behave similarly.



(a) MKL with (K_L-K_P).



(b) LMKL with (K_L-K_P).



(c) LMKL with (K_L-K_L-K_L).

Figure 3. Separating hyperplanes (black solid lines) and support vectors (filled points) on Banana data set. The gray solid lines shows the boundaries calculated from the gating models by considering them as classifiers which select a kernel function. Both accuracy increases and support vector count decreases.

We also combine $p = 2, \dots, 5$ linear kernels on benchmark data sets with LMKL. Table 1 (to the right) compares the results of canonical SVM with the linear kernel and LMKL with three linear kernels. LMKL uses statistically fewer support vectors on six out of ten data sets and on three of these (Banana, Pima, and Spambase), accuracy is significantly improved. With direct comparison of average values, LMKL performs better than canonical SVM on seven and eight out of ten data sets in terms of accuracy and support vector percentages, respectively. Using localized linear kernels also improves testing time due to evaluating linear kernels over only neighboring support vectors, instead of evaluating it over all support vectors.

Using Wilcoxon's signed rank test on ten data sets (see Table 1), when different kernels are combined, LMKL stores significantly fewer support vectors than MKL; when multiple copies of the same (linear) kernel are combined, LMKL achieves significantly higher accuracy than canonical SVM using a single kernel.

4.3. Bioinformatics Data Sets

We perform experiments on two bioinformatics data sets in order to see the applicability of LMKL to real-life problems. These translation initiation site data sets are constructed by using the same procedure described by Pedersen and Nielsen (1997). Each data instance is represented by a window of 200 nucleotides. Each nucleotide is encoded by five bits and the position of the set bit indicates whether the nucleotide is A, T, G, C, or N (for unknown).

As in benchmark data sets when combining different kernels, LMKL achieves statistically similar accuracy results compared with MKL by storing fewer support vectors for all combinations (see Table 2). For example, using (K_P-K_G), LMKL needs on the average 24.55 and 22.32 per cent fewer support vectors on Arabidopsis and Vertebrates data sets, respectively.

We combine $p = 2, \dots, 5$ linear kernels on bioinformatics data sets using LMKL. Table 2 shows that LMKL with three linear kernels improves the average accuracy statistically significantly. LMKL also uses significantly fewer support vectors (the decrease is almost one-third) on these data sets.

5. Conclusions

This work introduces a localized multiple kernel learning framework for kernel-based algorithms. The proposed algorithm consists of: (a) a gating model which assigns weights to kernels for a data instance, (b) a kernel-based learning algorithm with the locally combined kernel matrix. The training of these two components are coupled and the parameters of both components are optimized together by using a two-step alternate optimization procedure in a joint manner.

For binary classification tasks, the algorithm of the proposed framework with linear gating is derived and tested on ten benchmark and two bioinformatics data sets. LMKL achieves statistically similar accuracy results compared with MKL by storing fewer support vectors. Because kernels are evaluated locally (i.e., zero weighted kernels for a test instance are not calculated), the whole testing process is also much faster. This framework allows using multiple copies of the same kernel in different regions of the input space, obtaining more complex boundaries than what the underlying kernel is capable of. In order to illustrate this advantage, we combine different number of linear kernels on all data sets and learn piecewise linear boundaries. LMKL with three linear kernels gives significantly better accuracy results than canonical SVM with linear kernel on bioinformatics data sets.

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 Turkish Scientific Technical Research Council (TÜBİTAK) under Grant EEEAG 107E222. The work of M. Gönen was supported by the PhD scholarship (2211) from TÜBİTAK.

References

Alpaydın, E. (1999). Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, *11*, 1885–1892.

Amari, S., & Wu, S. (1998). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, *12*, 783–789.

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. *Proceedings of the 21st International Conference on Machine Learning* (pp. 41–48).

Collobert, R., Bengio, S., & Bengio, Y. (2001). A parallel mixture of SVMs for very large scale problems. *Advances in Neural Information Processing Systems (NIPS)* (pp. 633–640).

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton,

Table 1. The average testing accuracies and support vector percentages on benchmark data sets. Comparisons are performed between MKL and LMKL for (K_P-K_G). LMKL with (K_L-K_L-K_L) is compared to canonical SVM with K_L.

	SVM				MKL		LMKL		SVM		LMKL	
	K_P		K_G		(K_P-K_G)		(K_P-K_G)		K_L		(K_L-K_L-K_L)	
Data Set	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV
Banana	56.51	75.99	83.57	92.67	81.99	93.39	83.84	83.97	59.18	93.99	**81.39**	**54.03**
Germannumeric	71.80	54.17	68.65	58.44	73.32	84.89	73.92	80.90	74.58	97.09	75.09	**57.21**
Heart	72.78	73.89	77.67	79.11	75.78	87.89	79.44	**81.44**	78.33	67.00	77.00	**58.44**
Ionosphere	91.54	38.55	94.36	61.71	93.68	64.10	93.33	53.33	86.15	36.58	87.86	49.06
Liverdisorder	60.35	69.83	64.26	74.43	63.39	93.57	64.87	92.52	64.78	85.65	64.78	78.35
Pima	66.95	24.26	71.91	74.26	72.62	80.39	72.89	**73.63**	70.04	100.00	**73.98**	**53.09**
Ringnorm	70.66	53.91	98.82	40.68	98.86	57.68	98.69	56.69	76.91	78.68	78.92	**52.53**
Sonar	65.29	67.54	72.71	73.48	80.29	89.57	79.57	90.00	73.86	68.41	77.14	60.43
Spambase	84.18	47.92	79.80	49.50	90.46	57.47	91.41	58.24	85.98	77.43	**91.18**	**34.93**
Wdbc	88.73	27.11	94.44	54.74	95.50	58.11	95.98	**42.95**	95.08	**13.11**	94.34	21.89
5×2 cv Paired F Test (W-T-L)							0-10-0	3-7-0			3-7-0	6-4-0
Direct Comparison (W-T-L)							7-0-3	8-0-2			7-1-2	8-0-2
Wilcoxon's Signed Rank Test (W/T/L)							T	W			W	T

Table 2. The average testing accuracies and support vector percentages on bioinformatics data sets.

	SVM				MKL		LMKL		SVM		LMKL	
	K_P		K_G		(K_P-K_G)		(K_P-K_G)		K_L		(K_L-K_L-K_L)	
Data Set	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV
Arabidopsis	74.30	68.08	77.41	42.36	80.10	89.96	80.82	**65.41**	74.30	99.64	**81.29**	**68.66**
Vertebrates	75.50	68.54	75.72	41.64	78.67	90.46	77.67	68.14	75.50	99.02	**78.69**	**67.41**

G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87.

Lanckriet, G. R. G., Bie, T. D., Cristianini, N., Jordan, M. I., & Noble, W. S. (2004a). A statistical framework for genomic data fusion. *Bioinformatics*, *20*, 2626–2635.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004b). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Lee, W., Verzakov, S., & Duin, R. P. W. (2007). Kernel combination versus classifier combination. *Proceedings of the 7th International Workshop on Multiple Classifier Systems* (pp. 22–31).

Lewis, D. P., Jebara, T., & Noble, W. S. (2006). Nonstationary kernel combination. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 553–560).

Moguerza, J. M., Muñoz, A., & de Diego, I. M. (2004). Improving support vector classification via the combination of multiple sources of information. *Proceedings of Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops* (pp. 592–600).

Mosek (2008). *The MOSEK optimization tools manual version 5.0 (revision 79).* MOSEK ApS, Denmark.

Pavlidis, P., Weston, J., Cai, J., & Grundy, W. N. (2001). Gene functional classification from heterogeneous data. *Proceedings of the 5th Annual International Conference on Computational Molecular Biology* (pp. 242–248).

Pedersen, A. G., & Nielsen, H. (1997). Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology* (pp. 226–233).

Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. (2007). More efficiency in multiple kernel learning. *Proceedings of the 24th International Conference on Machine Learning* (pp. 775–782).

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.