To appear in 4th IAPR International Conference on Pattern Recognition in Bioinformatics PRIB 2009, Sheffield, 7-9 September, 2009

Multiple Kernel Machines Using Localized Kernels

Mehmet Gönen and Ethem Alpaydın

Department of Computer Engineering
Boğaziçi University
TR-34342, Bebek, İstanbul, Turkey
gonen@boun.edu.tr alpaydin@boun.edu.tr

Abstract. Multiple kernel learning (MKL) 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. Localized multiple kernel learning (LMKL) framework extends the MKL framework to allow combining kernels with different weights in different regions of the input space by using a gating model. LMKL extracts the relative importance of kernels in each region whereas MKL gives their relative importance over the whole input space. In this paper, we generalize the LMKL framework with a kernel-based gating model and derive the learning algorithm for binary classification. Empirical results on toy classification problems are used to illustrate the algorithm. Experiments on two bioinformatics data sets are performed to show that kernel machines can also be localized in a data-dependent way by using kernel values as gating model features. The localized variant achieves significantly higher accuracy on one of the bioinformatics data sets.

1 Introduction

In recent studies, different methods have been proposed for combining multiple kernels, instead of selecting a single one. The simplest approach is to use an unweighted sum of kernels [1]. This gives equal importance to each kernel and using a weighted sum (e.g., convex combination) is more reasonable. These weights also allows estimating the importance of kernels. The multiple kernel learning (MKL) framework [2, 3] uses an unweighted summation of discriminant values in different feature spaces which corresponds to a weighted summation of kernel values:

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

where m indexes kernels, $\boldsymbol{w}_m$ is the weight coefficients, $\Phi_m(\boldsymbol{x})$ is the mapping function for feature space m, and p is the number of kernels. After eliminating $\boldsymbol{w}_m$ from the model by using the duality conditions (see [3]), the discriminant

function becomes:

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

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

Using a fixed combination rule (unweighted or weighted) has the disadvantage of assigning the same weight to a kernel over the whole input space. If kernel weights can be assigned in a data-dependent way by considering the underlying localities in training data, a better learner may be produced. Lewis et al. [4] use a large-margin latent variable generative model for obtaining a nonstationary combination of kernels. Lee et al. [5] combine Gaussian kernels with different width parameters by forming a compositional kernel matrix to select suitable width for different regions of the input space. Gönen and Alpaydın [6] describe the localized multiple kernel learning (LMKL) framework which divides the input space into regions by using a parametric gating model and assigns higher combination weights to kernels which are suitable for each region.

In Sect. 2, we give a brief summary of the LMKL framework for binary classification problems [6] and then generalize it with a kernel-based gating model. We describe a generalized algorithm with a two-step alternate optimization method for hyperplane-based kernel machines using the localized kernel idea. Then in Sect. 3, we describe our experimental procedure and list our empirical results on toy and bioinformatics data sets. We summarize the results and conclude in Sect. 4.

2 Multiple Kernel Machines Using Localized Kernels

In this section, we give the derivations of localized kernel machines for binary classification. We describe the parametric gating model and also explain how we can use this model for non-vectorial data where we have the kernels but not necessarily a vectorial input. Then, we give the two-step alternate optimization method to train localized kernel machines.

2.1 Binary Classification

The LMKL framework divides the input space into regions and assigns combination weights to kernels in a data-dependent way. The discriminant function for binary classification is rewritten as:

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

where $\eta_m(\boldsymbol{x}|\boldsymbol{V})$ is a parametric gating model which assigns a weight to feature space m as a function of the input $\boldsymbol{x}$. Note that it is not required to use different

feature spaces, we can also use multiple copies of the same feature space in order to obtain a more complex discriminant function. For example, later on we will discuss using multiple linear kernels to define a piecewise linear boundary. By using (1) and regularizing the discriminant coefficients of all the feature spaces together, LMKL obtains the following optimization problem:

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

where C is the regularization parameter, $\boldsymbol{\xi}$ are the slack variables and $\boldsymbol{V}$ are the gating model parameters. The optimization problem in (2) is not convex due to the nonlinearity formed by using the gating model outputs in the separation constraints.

Suppose we know the gating model parameters, then, the model becomes convex and we can write the Lagrangian of the primal problem as:

$$\mathcal{L}_{\mathcal{C}} = \frac{1}{2}\sum_{m=1}^{p}\|\boldsymbol{w}_m\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i f_{\mathcal{C}}(\boldsymbol{x}_i) - 1 + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i$$

and taking the derivatives of $\mathcal{L}_{\mathcal{C}}$ with respect to the primal variables gives:

$$\begin{aligned} & \frac{\partial \mathcal{L}_{\mathcal{C}}}{\partial \boldsymbol{w}_m} \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^{n}\alpha_i y_i \eta_m(\boldsymbol{x}_i|\boldsymbol{V})\Phi_m(\boldsymbol{x}_i) \quad \forall m \\ & \frac{\partial \mathcal{L}_{\mathcal{C}}}{\partial b} \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & \frac{\partial \mathcal{L}_{\mathcal{C}}}{\partial \xi_i} \Rightarrow C = \alpha_i + \beta_i \quad \forall i \ . \end{aligned} \tag{3}$$

From $\mathcal{L}_{\mathcal{C}}$ and (3), the dual formulation is obtained as:

$$\begin{aligned} \max \ & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{w.r.t.} \ & \boldsymbol{\alpha} \\ \text{s.t.} \ & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i \end{aligned} \tag{4}$$

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

$$K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{p}\eta_m(\boldsymbol{x}_i|\boldsymbol{V})K_m(\boldsymbol{x}_i, \boldsymbol{x}_j)\eta_m(\boldsymbol{x}_j|\boldsymbol{V}) \ .$$

By using the gating model parameters and the support vector coefficients obtained from (4), we obtain the following discriminant function:

$$f_{\mathcal{C}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}) + b \ .$$

2.2 Gating Models

For gating, we can use different models for assigning kernel weights in a data-dependent way. Generally, we want to obtain sparse (with very few non-zero values) gating outputs for each data instance. This is usually achieved by using the softmax function at the output.

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

where $\boldsymbol{V} = \{\boldsymbol{v}_1, v_{10}, \boldsymbol{v}_2, v_{20}, \ldots, \boldsymbol{v}_p, v_{p0}\}$ and there are $p(d_{\mathcal{G}} + 1)$ parameters where $d_{\mathcal{G}}$ is the dimensionality of the gating feature space.

Gönen and Alpaydın [6] use a linear gating model which divides the input space into regions with linear boundaries. This corresponds to using $\boldsymbol{x}$ as $\boldsymbol{\Phi}_{\mathcal{G}}(\boldsymbol{x})$ in the gating model. If the linear gating model is not adequate for the problem at hand, we can use a more complex gating model by extracting additional features from the original features. For example, we can obtain quadratic boundaries for the gating model by concatenating the second order terms of the original features.

In some application areas such as bioinformatics, $\boldsymbol{x}$ vectors may appear in a non-vectorial format such as sequences, trees and graphs. In such a case where we can calculate K_m matrices but can not represent the data instances as $\boldsymbol{x}$ vectors, directly, we may define $\boldsymbol{\Phi}_{\mathcal{G}}(\boldsymbol{x})$ in terms of the kernel matrices:

$$\boldsymbol{\Phi}_{\mathcal{G}}(\boldsymbol{x}) = [K_{\mathcal{G}}(\boldsymbol{x}_1, \boldsymbol{x}) \ K_{\mathcal{G}}(\boldsymbol{x}_2, \boldsymbol{x}) \ \ldots \ K_{\mathcal{G}}(\boldsymbol{x}_n, \boldsymbol{x})]^T$$

where the gating kernel, $K_{\mathcal{G}}$, can be one of the combined kernels $K_1, K_2, \ldots, K_p$, a combination of them, or a completely different kernel used only for determining the gating boundaries. This gating model has number of parameters in the order of training instances and is not affected by the curse of dimensionality. This model has an advantage, if we have fewer training instances than their dimensionality (which is typically the case in bioinformatics and biometrics applications).

2.3 Training with Alternate Optimization

We can not perform the joint-optimization of the support vector coefficients and gating model parameters in (2), efficiently due to non-convexity. LMKL uses a two-step alternate optimization procedure in order to solve (2), as also used

for obtaining η_m parameters of MKL in a previous study [7]. This procedure consists of two basic steps: (a) solving the model with a fixed gating model and (b) updating the gating model parameters with the gradients calculated from the current solution.

Due to strong convexity, for a given $\boldsymbol{V}$, the gradients of the objective function in (2) is equal to the gradients of the objective function in (4). These gradients are found as:

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

where δ_m^k is 1 if $m = k$ and 0 otherwise. These gradients are used to update the gating model parameters at each step.

The complete algorithm for training localized kernel machines is summarized in Algorithm 1. Step size, μ, is optimized using Armijo's rule and this guarantees to get a better objective value than the previous one at each step. Note however that this does not mean that we obtain the global optimum as a result of this procedure. We can get stuck at a local optimum and the starting point (i.e., initial gating model) may affect the solution quality.

Algorithm 1 Multiple Kernel Machines using Localized Kernels

1: Initialize $\boldsymbol{V}$ to small random numbers
2: **repeat**
3: Calculate $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$ using $\boldsymbol{V}$
4: Solve kernel machine with $K_\eta(\boldsymbol{x}_i,\boldsymbol{x}_j)$
5: Determine μ using Armijo's rule
6: $\boldsymbol{V} \Leftarrow \boldsymbol{V} - \mu\dfrac{\partial J_\eta}{\partial \boldsymbol{V}}$
7: **until** convergence

Any kernel machine which has a hyperplane-based decision function can also be localized by replacing $\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x})\rangle$ with $\sum_{m=1}^{p}\eta_m(\boldsymbol{x}|\boldsymbol{V})\langle \boldsymbol{w}_m, \boldsymbol{\Phi}_m(\boldsymbol{x})\rangle$ and deriving the corresponding update rules.

3 Experiments

We implement Algorithm 1 in C++ and solve the optimization problems for canonical kernel machines with MOSEK optimization software [8]. We perform

experiments as follows: For a given data set, we select one-third randomly as the test set and generate ten training and validation sets from the remaining two-thirds by using 5×2 cross-validation (with stratification for classification problems). The validation sets are used to select the best C from the set {0.01, 0.1, 1, 10, 100}. The configuration which has the highest average validation performance is trained on ten different training folds and the performance is measured over the test set. We have ten test set results for each data set. Linear (K_L) and polynomial (K_P) kernels are used on toy data sets:

$$K_L(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
$$K_P(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^q \ .$$

All kernel matrices are calculated and normalized to unit trace before training for classification problems.

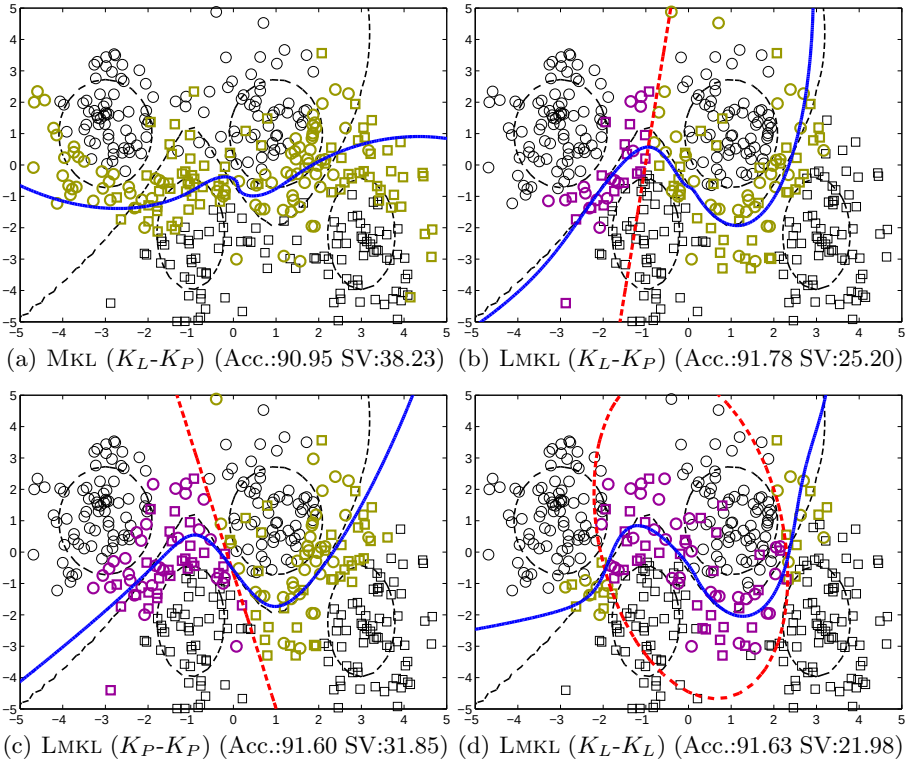
3.1 Toy Classification Data Sets

In order to illustrate the method on binary classification problems, we use the toy data set named GAUSS4 which is taken from Gönen and Alpaydın [6] which consists of 1200 data instances generated from four Gaussian components (two for each class) with the following proportions, mean vectors and covariance matrices:

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

where data instances from the first three components are of class 1 (labeled as positive) and others are of class 2 (labeled as negative). Figure 1 shows the classification boundaries and the support vectors selected in these experiments. We train MKL with (K_L-K_P) and LMKL for the following combinations: (K_L-K_P) with linear gating, (K_P-K_P) with linear gating, and (K_L-K_L) with quadratic gating. For the quadratic gating model, we simply add the second order terms of the original features in $\Phi_{\mathcal{G}}(\boldsymbol{x})$ and we use second order ($q = 2$) polynomial kernel as K_P.

The complexity of the obtained classification boundaries are controlled by two main factors: (i) the complexity of combined kernels and (ii) the gating model complexity. For example, (K_L-K_P) with linear gating and (K_L-K_L) with quadratic gating produce similar approximations of the optimal Bayes' boundary (Fig. 1(b), (c)). If we use simple kernels such as the linear kernel in combination, we may need a more complex gating model to obtain good boundaries.



(a) MKL (K_L-K_P) (Acc.:90.95 SV:38.23) (b) LMKL (K_L-K_P) (Acc.:91.78 SV:25.20)

(c) LMKL (K_P-K_P) (Acc.:91.60 SV:31.85) (d) LMKL (K_L-K_L) (Acc.:91.63 SV:21.98)

Fig. 1. Fitted functions (solid lines) and support vectors (bold points) on GAUSS4 data set. Dashed lines show the Gaussians from which data are sampled and the optimal Bayes' discriminant. The thick dashed lines show the gating boundaries (where $\eta_i(\boldsymbol{x}) = \eta_j(\boldsymbol{x})$ and i, j are neighboring kernels) and the thick lines show the learned class boundaries. Test accuracies and stored support vector percentages are given. Note that the softmax function allows a smooth transition between regions.

Selecting the best kernel function for a given data set is generally performed by using a statistical cross-validation procedure. Modifying kernel combination rule with a localized kernel has the advantage of choosing the required complexity automatically for the task at hand. For example, Fig. 2 shows the average testing errors and support vector percentages of LMKL with multiple copies of the linear kernel on GAUSS4, we have similar error rates after 3 components which gives the underlying complexity of this classification problem. LMKL uses 3 linear kernels actively by assigning zero gating outputs to some of the kernels with the help of the softmax function. It also stores nearly the same number of support vectors after this point. This indicates that even if we start with more kernels than necessary, LMKL automatically regularizes and does not overfit. Regularizing $\boldsymbol{w}_m$ terms given in (3) also enforces using fewer non-zero gating outputs as well as fewer support vectors.

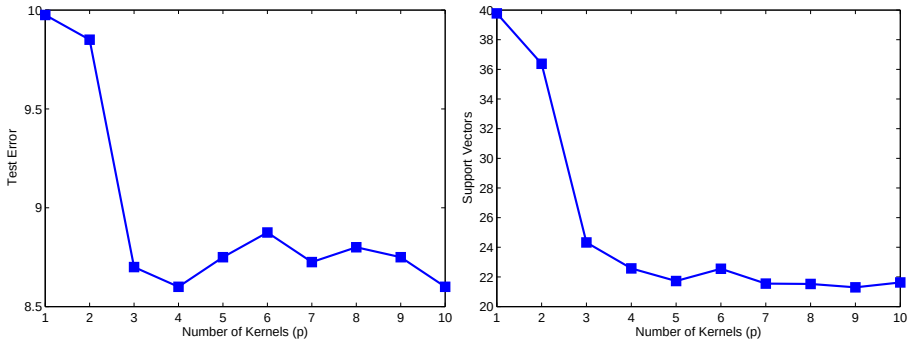


Fig. 2. The average testing errors and support vector percentages with $p = 1, \ldots, 10$ linear kernels on GAUSS4 data set.

3.2 Bioinformatics Data Sets

We perform protein location prediction experiments on the MIPS Comprehensive Yeast Genome Database (CYGD) [9]. CYGD assigns subcellular locations for 2318 and 1150 proteins according to whether they participate in the membrane and the ribosome, respectively. For this problem, we have different similarity measures between proteins but we do not have the exact vectorial representations of data instances. In order to learn the gating model, pairwise function values of one kernel is fed into the gating model as features. In the experiments, we use the three kernel functions in Table 1. The results of single-kernel SVMs, MKL and LMKL using each kernel alone in the gating model are given in Table 2.

Table 1. Kernels used for protein location prediction problem from [10].

Kernel	Explanation
K_1	Gaussian kernel calculated from gene expression
K_2	Linear kernel calculated from protein interactions
K_3	Smith-Waterman kernel calculated from protein sequences

On MEMBRANE, LMKL with K_2 or K_3 used in the gating model achieves significantly higher accuracy than MKL and single-kernel SVMs. All LMKL variants, MKL and single-kernel SVMs achieve comparable accuracies on RIBOSOMAL. In Table 3, the weights found by MKL for the three kernels are given; we see that on MEMBRANE, MKL uses K_1 and K_3; on RIBOSOMAL, it heavily uses K_1. In the same table, we also report the sum of $\eta_m(\boldsymbol{x}), m = 1, 2, 3$ of the support vectors normalized by dividing by the number of support vectors. We see that on MEMBRANE, LMKL uses K_2 and K_3; on RIBOSOMAL, with K_1 or K_3 in the gating model, LMKL also uses K_1 heavily; otherwise, it tends to use all three. This indicates that LMKL also allows knowledge extraction in that we can see

Table 2. The average testing accuracies and support vector percentages of single-kernel SVM, MKL and LMKL with the kernel-based gating model on protein location prediction experiments using (K_1-K_2-K_3) combination. An underlined entry means that LMKL is significantly better than single-kernel SVMs and MKL.

	SVM						MKL		LMKL					
	K_1		K_2		K_3				K_1		K_2		K_3	
	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV
M	84.79	72.93	81.19	86.67	78.94	75.29	84.68	77.24	84.49	74.42	<u>86.07</u>	73.81	<u>85.34</u>	83.21
R	95.26	46.21	92.16	81.41	98.70	17.96	98.70	18.17	98.65	36.97	97.19	24.62	98.46	30.22

which kernel returns the most useful similarity metric for which part of the input space.

Table 3. The average weights assigned to kernels by MKL and LMKL with the kernel-based gating model on protein location prediction experiments using (K_1-K_2-K_3) combination.

	MKL	LMKL		
		K_1	K_2	K_3
	(η_1-η_2-η_3)	(η_1-η_2-η_3)	(η_1-η_2-η_3)	(η_1-η_2-η_3)
MEMBRANE	0.25-0.02-0.73	0.09-0.42-0.49	0.00-0.66-0.34	0.17-0.41-0.42
RIBOSOMAL	0.96-0.02-0.01	0.97-0.03-0.00	0.38-0.27-0.35	0.99-0.01-0.00

4 Conclusion

We generalize the localized multiple kernel learning framework with a kernel-based gating model and give the learning algorithm for binary classification. This framework also allows us to determine the model complexity at the combination level instead of performing statistical cross-validation. The algorithm for binary classification is illustrated on toy data sets.

Localized kernel machines are tested on two bioinformatics data sets and we use kernel values as gating model features in these experiments. We achieve significantly higher accuracy on one of the bioinformatics data sets by combining kernels in a data-dependent way.

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, the 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

1. Pavlidis, P., Weston, J., Cai, J., Grundy, W.N.: Gene functional classification from heterogeneous data. In: Proceedings of the 5th Annual International Conference on Computational Molecular Biology. (2001) 242–248
2. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research **5** (2004) 27–72
3. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st International Conference on Machine Learning. (2004) 41–48
4. Lewis, D.P., Jebara, T., Noble, W.S.: Nonstationary kernel combination. In: Proceedings of the 23rd International Conference on Machine Learning. (2006) 553–560
5. Lee, W., Verzakov, S., Duin, R.P.W.: Kernel combination versus classifier combination. In: Proceedings of the 7th International Workshop on Multiple Classifier Systems. (2007) 22–31
6. Gönen, M., Alpaydın, E.: Localized multiple kernel learning. In: Proceedings of the 25th International Conference on Machine Learning. (2008) 352–359
7. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: More efficiency in multiple kernel learning. In: Proceedings of the 24th International Conference on Machine Learning. (2007) 775–782
8. Mosek: The MOSEK Optimization Tools Manual Version 5.0 (Revision 105). MOSEK ApS, Denmark. (2009)
9. Mewes, H.W., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schller, C., Stocker, S., Weil, B.: MIPS: A database for genomes and protein sequences. Nucleic Acid Research **28** (2000) 37–40
10. Lanckriet, G.R.G., Bie, T.D., Cristianini, N., Jordan, M.I., Noble, W.S.: A statistical framework for genomic data fusion. Bioinformatics **20**(16) (2004) 2626–2635