# Multiple Kernel Learning Algorithms

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

## Abstract

In recent years, several methods have been proposed to combine multiple kernels instead of using a single one. These different kernels may correspond to using different notions of similarity or may be using information coming from multiple sources (different representations or different feature subsets). In trying to organize and highlight the similarities and differences between them, we give a taxonomy of and review several multiple kernel methods. We perform experiments for better illustration. We see that though there may not be large differences in terms of accuracy, there is difference between them in complexity as given by the number of stored support vectors, the sparsity of the solution as given by the number of used kernels, and training time.

**Keywords:** Support vector machines, kernel machines, multiple kernel learning

## Notation

| | |
|---|---|
| $\mathbb{R}$ | Real numbers |
| $\mathbb{R}_+$ | Nonnegative real numbers |
| $\mathbb{R}_{++}$ | Positive real numbers |
| $\mathbb{R}^N$ | Real $N \times 1$ matrices |
| $\mathbb{R}^{M \times N}$ | Real $M \times N$ matrices |
| $\mathbb{N}$ | Natural numbers |
| $\mathbb{S}^N$ | Symmetric $N \times N$ matrices |
| $\left\|\boldsymbol{x}\right\|_p$ | $l_p$-norm of vector $\boldsymbol{x}$ |
| $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ | Dot product between $\boldsymbol{x}$ and $\boldsymbol{y}$ |
| $k(\boldsymbol{x}, \boldsymbol{y})$ | Kernel function |
| $\mathbf{K}$ | Kernel matrix |
| $\mathbf{X}^T$ | Transpose of matrix $\mathbf{X}$ |
| $\mathrm{tr}\left(\mathbf{X}\right)$ | Trace of matrix $\mathbf{X}$ |
| $\left\|\mathbf{X}\right\|_F$ | Frobenious norm of matrix $\mathbf{X}$ |
| $\mathbf{X} \odot \mathbf{Y}$ | Element-wise product between $\mathbf{X}$ and $\mathbf{Y}$ |

## 1. Introduction

Support vector machine (Svm) is a discriminative classifier proposed for binary classification problems and is based on the theory of structural risk minimization (Vapnik, 1998). Given a sample of $N$ independent and identically distributed training instances $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where $\boldsymbol{x}_i$ is the $D$-dimensional input vector and $y_i \in \{-1, +1\}$ is its class label, Svm basically finds the linear discriminant with the maximum margin in the feature space induced by the mapping function $\Phi$. The resulting discriminant function is:

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

The classifier can be trained by solving the following quadratic optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \left\|\boldsymbol{w}\right\|_2^2 + C \sum_{i=1}^N \xi_i \\ \text{with respect to} \quad & \boldsymbol{w} \in \mathbb{R}^D, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \quad & y_i \left( \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i \end{aligned}$$

where $\boldsymbol{w}$ is the vector of weight coefficients, $C$ is a predefined positive trade-off parameter between model simplicity and classification error, $\boldsymbol{\xi}$ is the vector of slack variables, and $b$ is the bias term of the separating hyperplane. Instead of solving this optimization problem directly, the Lagrangian dual function enables us to obtain the following dual formulation:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \overbrace{\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle}^{k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)} \\ \text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \end{aligned} \tag{1}$$

where $\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle$ is named as *kernel function* $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ and $\boldsymbol{\alpha}$ is the vector of dual variables corresponding to each separation constraint. Solving this, we get $\boldsymbol{w} = \sum_{i=1}^N \alpha_i y_i \Phi(\boldsymbol{x}_i)$ and the discriminant function can be written as:

$$f(x) = \sum_{i=1}^N \alpha_i y_i k\left(\boldsymbol{x}_i, \boldsymbol{x}\right) + b \, .$$

There are several kernel functions successfully used in the literature such as linear kernel ($k_L$), polynomial kernel ($k_P$), and Gaussian kernel ($k_G$):

$$\begin{aligned} k_L\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) &= \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ k_P\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) &= \left( \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1 \right)^q \qquad q \in \mathbb{N} \\ k_G\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) &= \exp\left( - \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2 / s^2 \right) \qquad s \in \mathbb{R}_{++} \, . \end{aligned}$$

There are also kernel functions proposed for particular applications, such as natural language processing (Lodhi et al., 2002) and bioinformatics (Schölkopf et al., 2004).

Kernel machines generally learn a decision function in terms of kernel values between training instances, $\{\boldsymbol{x}_i\}_{i=1}^N$, and the test instance, $\boldsymbol{x}$, as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \,.$$

Selecting the kernel function $k$ and its parameters (e.g., $q$ or $s$) is an important issue in training. Generally, a cross-validation procedure is used to choose the best performing kernel function among a set of kernel functions on a separate validation set different from the training set. In recent years, multiple kernel learning (MKL) methods have been proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_\eta\left(k_1(\boldsymbol{x}_i, \boldsymbol{x}_j), k_2(\boldsymbol{x}_i, \boldsymbol{x}_j), \ldots, k_P(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \tag{2}$$

where the combination function, $f_\eta$, can be a linear or a nonlinear function of the input kernels. The reasoning is similar to combining different classifiers: Instead of choosing a single kernel function and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking or combination.

There can be two uses of multiple kernel learning:

(a) Different kernels correspond to different notions of similarity and instead of trying to find which works best, a learning method does the picking for us, or may use a combination of them. Using a specific kernel may be a source of bias and in allowing a learner to choose among a set of kernels, a better solution can be found.

(b) Different kernels may be using input coming from different input representations possibly from different input sources or modalities. Since these are different representations, they have different measures of similarity and therefore use different kernels. In such a case, combining kernels is one possible way to combine multiple inputs. Schölkopf et al. (2004, chap. 3) call this method of combining kernels *intermediate combination* and contrasts this with *early combination* (where features from different sources are concatenated and fed to a single learner) and *late combination* (where different features are fed to different classifiers whose decisions are then combined by a fixed or trained combiner).

There are a significant number of work in the literature for combining multiple kernels. We give a taxonomy in Table 1 and the following sections follow this taxonomy. We give experimental results in Section 7 and conclude in Section 8.

## 2. Fixed Rules for Kernel Combination

Fixed rules for kernel combination use $f_\eta$ in (2) as a fixed function of the kernels, without any training. Once we calculate $k_\eta$ using $f_\eta$, we train a canonical kernel machine with $k_\eta$. For example, we can obtain a valid kernel by taking summation or multiplication of two valid kernels as follows (Cristianini and Shawe-Taylor, 2000):

$$\begin{aligned} k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + k_2(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= k_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\, k_2(\boldsymbol{x}_i, \boldsymbol{x}_j) \;. \end{aligned} \tag{3}$$



Table 1: Taxonomy of multiple kernel learning algorithms.

| |
|---|
| FIXED RULES |
| Pavlidis et al. (2001); Ben-Hur and Noble (2005) |
| PARAMETERIZED FUNCTIONS |
| **Linear Functions** |
| *Linear Combination* |
| Lanckriet et al. (2002, 2004b); Conforti and Guido (2009) |
| *Nonnegative Linear Combination* |
| Fung et al. (2004); Lanckriet et al. (2004b); Tsuda et al. (2004); Qiu and La (2005); Lin et al. (2009); Zhao et al. (2009); Kloft et al. (2010) |
| *Convex Combination* |
| Bousquet and Herrmann (2003); Bach et al. (2004); Argyriou et al. (2005, 200 Micchelli and Pontil (2005); Kim et al. (2006); Sonnenburg et al. (2006a,b); De et al. (2007); Rakotomamonjy et al. (2007, 2008); Ye et al. (2007a); Zien and C (2007); Longworth and Gales (2008, 2009); Xu et al. (2009a) |
| *Binary Combination* |
| Xu et al. (2009b) |
| **Nonlinear Functions** |
| Ong et al. (2003, 2005); Ong and Smola (2003); Lewis et al. (2006b); Varma and Babu (2007, 2009); Gönen and Alpaydın (2008, 2009); Cortes et al. (2010) |
| SIMILARITY-BASED METHODS |
| **Kernel Alignment** |
| *Linear Combination* |
| Lanckriet et al. (2004b); Igel et al. (2007) |
| *Nonnegative Linear Combination* |
| Kandola et al. (2002); Lanckriet et al. (2004b) |
| *Convex Combination* |
| Qiu and Lane (2009) |
| **Other Similarity Measures** |
| He et al. (2008); Nguyen and Ho (2008); Ying et al. (2009) |
| BOOSTING METHODS |
| Bennett et al. (2002); Crammer et al. (2003); Bi et al. (2004) |
| BAYESIAN METHODS |
| Girolami and Rogers (2005); Girolami and Zhong (2007); Damoulas and Girolami (2008, 2009a,b) |

We can apply the rules in (3) recursively to obtain the rules for more than two kernels. For example, the weighted summation of $P$ kernels is also a valid kernel:

$$\begin{aligned} k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{m=1}^{P} k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \prod_{m=1}^{P} k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \;. \end{aligned} \tag{4}$$

Pavlidis et al. (2001) report that on a gene functional classification task, training an SVM with an unweighted sum of heterogeneous kernels gives better results than the combination of multiple SVMs each trained with one of these kernels.

Pairwise kernels are used to express the similarity between pairs (e.g., of proteins in terms of similarities between individual proteins). The simplest way to define a pairwise kernel is:

$$k^P\left((\boldsymbol{x}_i^a,\boldsymbol{x}_j^a),(\boldsymbol{x}_i^b,\boldsymbol{x}_j^b)\right)=k(\boldsymbol{x}_i^a,\boldsymbol{x}_i^b)k(\boldsymbol{x}_j^a,\boldsymbol{x}_j^b)+k(\boldsymbol{x}_i^a,\boldsymbol{x}_j^b)k(\boldsymbol{x}_j^a,\boldsymbol{x}_i^b)\;.$$

Ben-Hur and Noble (2005) combine pairwise kernels in two different ways:

$$k_\eta^P\left((\boldsymbol{x}_i^a,\boldsymbol{x}_j^a),(\boldsymbol{x}_i^b,\boldsymbol{x}_j^b)\right)=\sum_{m=1}^{P}k_m^P\left((\boldsymbol{x}_i^a,\boldsymbol{x}_j^a),(\boldsymbol{x}_i^b,\boldsymbol{x}_j^b)\right)$$
$$k_\eta^P\left((\boldsymbol{x}_i^a,\boldsymbol{x}_j^a),(\boldsymbol{x}_i^b,\boldsymbol{x}_j^b)\right)=k_\eta(\boldsymbol{x}_i^a,\boldsymbol{x}_i^b)k_\eta(\boldsymbol{x}_j^a,\boldsymbol{x}_j^b)+k_\eta(\boldsymbol{x}_i^a,\boldsymbol{x}_j^b)k_\eta(\boldsymbol{x}_j^a,\boldsymbol{x}_i^b)$$

and improve the classification performance for protein-protein interaction prediction task.

## 3. Learning Parameterized Functions of Kernels

In (2), instead of using a fixed $f_\eta$, we can have a function parameterized by a set of parameters $\theta$ and then we have a learning procedure to optimize $\theta$ as well.

### 3.1 Linear Functions of Kernels

The simplest case is to parameterize the sum rule as:

$$k_\eta\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=f_\eta\left(k_1\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right),k_2\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right),\ldots,k_P\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)|\theta=\boldsymbol{\eta}\right)=\sum_{m=1}^{P}\eta_m k_m\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)\;.$$

Different versions of this approach differ in the way they put restrictions on $\boldsymbol{\eta}$.

#### 3.1.1 Linear Combination

Lanckriet et al. (2002, 2004b) follow a direct approach in order to optimize the unrestricted kernel combination weights. The implausibility of a kernel, $\omega(\mathbf{K})$, is defined as the objective function value obtained after solving a canonical SVM optimization problem (Here we only consider the soft margin formulation, which uses $l_1$-norm on slack variables).

$$\begin{aligned}\text{maximize}\;\;&\omega(\mathbf{K})=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)\\ \text{with respect to}\;\;&\boldsymbol{\alpha}\in\mathbb{R}_+^N\\ \text{subject to}\;\;&\sum_{i=1}^{N}\alpha_i y_i=0\\ &C\geq\alpha_i\geq 0\qquad\forall i\end{aligned}$$



The combined kernel matrix is selected from the following set:

$$\mathcal{K}_L=\left\{\mathbf{K}:\mathbf{K}=\sum_{m=1}^{P}\eta_m\mathbf{K}_m,\;\;\mathbf{K}\succeq 0,\;\;\mathrm{tr}\left(\mathbf{K}\right)\leq c\right\}$$

where the selected kernel is forced to be positive semidefinite.

The resulting optimization problem that minimizes the implausibility of the combined kernel matrix (the objective function value of the corresponding soft margin SVM optimization problem) is formulated as:

$$\begin{aligned}\text{minimize}\;\;&\omega(\mathbf{K}_\eta^{\mathrm{tra}})\\ \text{with respect to}\;\;&\mathbf{K}_\eta\in\mathcal{K}_L\\ \text{subject to}\;\;&\mathrm{tr}\left(\mathbf{K}_\eta\right)=c\end{aligned}$$

where $\mathbf{K}_\eta^{\mathrm{tra}}$ is the kernel matrix calculated over only the training set and this problem can be cast into the following SDP formulation:

$$\begin{aligned}\text{minimize}\;\;&t\\ \text{with respect to}\;\;&\boldsymbol{\eta}\in\mathbb{R}^P,t\in\mathbb{R},\lambda\in\mathbb{R},\boldsymbol{\nu}\in\mathbb{R}_+^N,\boldsymbol{\delta}\in\mathbb{R}_+^N\\ \text{subject to}\;\;&\mathrm{tr}\left(\mathbf{K}_\eta\right)=c\\ &\begin{pmatrix}\left(\boldsymbol{y}\boldsymbol{y}^T\right)\odot\mathbf{K}_\eta^{\mathrm{tra}} & \boldsymbol{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\boldsymbol{y}\\ \left(\boldsymbol{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\boldsymbol{y}\right)^T & t-2C\boldsymbol{\delta}^T\boldsymbol{e}\end{pmatrix}\succeq 0\\ &\mathbf{K}_\eta\succeq 0\;.\end{aligned}$$

This optimization problem is defined for a transductive learning setting and we need to be able to calculate the kernel function values for test instances as well as training instances.

Lanckriet et al. (2004a,b) considers predicting function classifications associated with yeast proteins. Different kernels calculated on heterogeneous genomic data, namely, amino acid sequences, protein-protein interactions, genetic interactions, protein complex data, and expression data, are combined using SDP formulation. This gives better results than SVMs trained with each kernel in nine out of 13 experiments. Conforti and Guido (2009) propose another SDP formulation that removes trace restriction on the combined kernel matrix and introduces constraints over the kernel weights for an inductive setting.

#### 3.1.2 Nonnegative Linear Combination

Lanckriet et al. (2004b) restrict the combination weights to have nonnegative values by selecting the combined kernel matrix from:

$$\mathcal{K}_P=\left\{\mathbf{K}:\mathbf{K}=\sum_{m=1}^{P}\eta_m\mathbf{K}_m,\;\;\boldsymbol{\eta}\geq 0,\;\;\mathbf{K}\succeq 0,\;\;\mathrm{tr}\left(\mathbf{K}\right)\leq c\right\}\;.$$

and reduce the SDP formulation to the following QCQP optimization problem by selecting the combined kernel matrix from $\mathcal{K}_P$ instead of $\mathcal{K}_L$:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}ct - \sum_{i=1}^{N}\alpha_i \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N, t \in \mathbb{R} \\
\text{subject to} \quad & \operatorname{tr}(\mathbf{K}_m)\, t \geq \boldsymbol{\alpha}^T \left(\left(\boldsymbol{y}\boldsymbol{y}^T\right) \odot \mathbf{K}_m^{\text{tra}}\right) \boldsymbol{\alpha} \qquad \forall m \\
& \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned}$$

where we can jointly find the support vector coefficients and the kernel combination weights. This optimization problem is also developed for a transductive setting, but we can simply take the number of test instances as zero and find the kernel combination weights for an inductive setting. The interior-point methods used to solve this QCQP formulation also return the optimal values of the dual variables that correspond to the optimal kernel weights. Qiu and Lane (2005) give SILP and QCQP formulations of regression estimation using $\epsilon$-tube SVR. The QCQP formulation is used for predicting siRNA efficacy by combining kernels over heterogeneous data sources (Qiu and Lane, 2009). Zhao et al. (2009) develop a multiple kernel learning method for clustering problems using the maximum margin clustering idea of Xu et al. (2005) and a nonnegative linear combination of kernels.

Lanckriet et al. (2004b) combine two different kernels obtained from heterogeneous information sources, namely, bag-of-words and graphical representations, on Reuters-21578 data set. Combining these two kernels with positive weights outperforms single kernel results obtained with SVM on four tasks out of five. Lanckriet et al. (2004c) uses QCQP formulation to integrate multiple kernel functions calculated on heterogeneous views of genome data obtained through different experimental procedures. These views include amino acid sequences, hydropathy profiles, gene expression data and known protein-protein interactions. The prediction task is to recognize the particular classes of proteins, namely, membrane proteins and ribosomal proteins. QCQP approach gives significantly better results than any single kernel and the unweighted sum of kernels. The assigned kernel weights also enable us to extract the relative importance of the data sources feeding the separate kernels. This approach assigns near zero weights to random kernels added to the candidate set of kernels before training. Dehak et al. (2008) combine three different kernels obtained on the same features and get better results than score fusion for speaker verification problem.

Fung et al. (2004) propose an iterative algorithm based on kernel Fisher discriminant analysis to combine heterogeneous kernel in a linear manner with nonnegative weights. The proposed method requires solving a simple nonsingular system of linear equations of size $(N+1)$ and a QP problem having $P$ decision variables at each iteration. On a colorectal cancer diagnosis task, this method obtains similar results using much less computation time compared to selecting a kernel for standard kernel Fisher discriminant analysis.

Tsuda et al. (2004) learn the kernel combination weights by minimizing an approximation of the cross-validation error for kernel Fisher discriminant analysis. In order to update the kernel combination weights, cross-validation error should be approximated with



a differentiable error function. Tsuda et al. (2004) use the sigmoid function for approximation and derive the update rules of the kernel weights. This procedure requires inverting a $N \times N$ matrix and calculating the gradients at each step. Tsuda et al. (2004) combine heterogeneous data sources using kernels, which are mixed linearly and nonlinearly, for bacteria classification and gene function prediction tasks. Fisher discriminant analysis with the combined kernel matrix, which is optimized using the cross-validation error approximation, gives significantly better results than single kernels for both tasks.

In order to consider the capacity of the resulting classifier, Tan and Wang (2004) optimize the nonnegative combination coefficients by using the minimal upper bound of the Vapnik-Chervonenkis dimension as the target function.

Lin et al. (2009) propose a dimensionality reduction method that uses multiple kernels to embed data instances from different feature spaces to a unified feature space. The method is derived from a graph embedding framework using kernel matrices instead of data matrices. The learning phase is performed using a two-step alternate optimization procedure that updates the dimensionality reduction coefficients and the kernel weights in turn. McFee and Lanckriet (2009) propose a method for learning a unified space from multiple kernels calculated over heterogeneous data sources. This method uses a partial order over pairwise distances as the input and produces an embedding using graph-theoretic tools. The kernel (data source) combination rule is learned by solving an SDP problem and all input instances are mapped to the constructed common embedding space.

Kloft et al. (2010) generalize the MKL formulation for arbitrary $l_p$ ($p \geq 1$) norms by regularizing over the kernel coefficients (which is done by adding $\mu \|\boldsymbol{\eta}\|_p^p$ to the objective function) or equivalently constraining them ($\|\boldsymbol{\eta}\|_p^p \leq 1$). The resulting optimization problem is:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\left(\sum_{m=1}^{P}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_i y_i y_j k_m\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\right)^{\frac{p-1}{p}}\right)^{\frac{p}{p-1}} \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i\ .
\end{aligned}$$

and Kloft et al. (2010) solve this problem by using alternative optimization strategies based on Newton descent and cutting planes.

Lewis et al. (2006a) compare the performances of unweighted and weighted sums of kernels on a gene functional classification task. Their results can be summarized with two guidelines: (a) When all kernels or data sources are informative, we should use the unweighted sum rule. (b) When some of the kernels or the data sources are noisy or irrelevant, we should optimize the kernel weights.

Usually, the kernel weights are constrained by a trace or $l_1$-norm regularization. Cortes et al. (2009) discuss the suitability of $l_2$-norm for MKL. They combine kernels with ridge

regression using $l_2$-norm regularization over the kernel weights. They conclude that using $l_1$-norm improves the performance for a small number of kernels, but degrades the performance when combining a large number of kernels. However, $l_2$-norm never decreases the performance and significantly increases the performance for larger sets of candidate kernels. Yan et al. (2009) compare $l_1$-norm and $l_2$-norm for image and video classification tasks, and conclude that $l_2$-norm should be used when the combined kernels carry complementary information.

#### 3.1.3 Convex Combination

We can think of kernel combination as a weighted average of kernels and consider $\boldsymbol{\eta} \in \mathbb{R}_+^P$ and $\sum_{m=1}^P \eta_m = 1$. Joachims et al. (2001) show that combining two kernels is beneficial if both of them achieve approximately the same performance and use different data instances as support vectors. This makes sense: In combination, we want kernels to be useful by themselves and complementary. In a web page classification experiment, combining the word and the hyperlink representations through the convex combination of two kernels (i.e., $\eta_2 = 1 - \eta_1$) can achieve better classification accuracy than each of the kernels.

Chapelle et al. (2002) calculate the derivative of the margin and the derivative of the radius (of the smallest sphere enclosing the training points) with respect to a kernel parameter, $\theta$:

$$\frac{\partial \left\|\boldsymbol{w}\right\|_2^2}{\partial \theta} = -\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \frac{\partial k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)}{\partial \theta}$$

$$\frac{\partial R^2}{\partial \theta} = \sum_{i=1}^N \beta_i \frac{\partial k\left(\boldsymbol{x}_i, \boldsymbol{x}_i\right)}{\partial \theta} - \sum_{i=1}^N \sum_{j=1}^N \beta_i \beta_j \frac{\partial k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)}{\partial \theta}$$

where $\boldsymbol{\alpha}$ is obtained by solving the canonical Svm optimization problem and $\boldsymbol{\beta}$ is obtained by solving the quadratic programming (Qp) problem defined by Vapnik (1998). These derivatives can be used to optimize the individual parameters (e.g., scaling coefficient) on each feature by using an alternating optimization procedure (Weston et al., 2001; Chapelle et al., 2002; Grandvalet and Canu, 2003). This strategy is also a multiple kernel learning approach because the optimized parameters can be interpreted as the kernel parameters, and then we combine these kernel values over all features.

Bousquet and Herrmann (2003) rewrite the gradient of the margin by replacing $\mathbf{K}$ with $\mathbf{K}_\eta$ and taking the derivative with respect to the kernel weights gives:

$$\frac{\partial \left\|\boldsymbol{w}_\eta\right\|_2^2}{\partial \eta_m} = -\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \frac{\partial k_\eta\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)}{\partial \eta_m} = -\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k_m\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \qquad \forall m$$

where $\boldsymbol{w}_\eta$ is the weight vector obtained using $\mathbf{K}_\eta$ in training. In an iterative manner, an Svm is trained to obtain $\boldsymbol{\alpha}$, then $\boldsymbol{\eta}$ is updated using the calculated gradient while considering nonnegativity (i.e., $\boldsymbol{\eta} \in \mathbb{R}_+^P$) and normalization (i.e., $\sum_{m=1}^P \eta_m = 1$). This procedure considers the performance (in terms of margin maximization) of the resulting classifier, which uses the combined kernel matrix.



Micchelli and Pontil (2005) try to learn an optimal kernel over the convex hull of predefined basic kernels by minimizing a regularization functional. Their analysis shows that any optimizing kernel can be expressed as the convex combination of basic kernels. Argyriou et al. (2005, 2006) build practical algorithms for learning a suboptimal kernel when the basic kernels are continuously parameterized by a compact set. This continuous parameterization allows us to select kernels from basically an infinite set instead of a finite number of basic kernels.

Kim et al. (2006) show that selecting the optimal kernel from the set of convex combinations over the candidate kernels can be formulated as a convex optimization problem. This formulation is more efficient than the iterative approach of Fung et al. (2004). Ye et al. (2007a) formulate a semidefinite programming (Sdp) problem inspired by Kim et al. (2006) for learning an optimal kernel over a convex set of candidate kernels for regularized kernel discriminant analysis. The Sdp formulation can be modified so that it can jointly optimize the kernel weights and the regularization parameter. Ye et al. (2007b, 2008) derive the quadratically constrained quadratic programming (Qcqp) and semi-infinite linear programming (Silp) formulations equivalent to the previous Sdp problem in order to reduce the time complexity. These three formulations are directly applicable to multiclass classification due to being dependent on regularized kernel discriminant analysis.

De Bie et al. (2007) derive Qcqp formulation of one-class classification problem using a convex combination of multiple kernels. In order to prevent the combined kernel from overfitting, they also propose a modified mathematical model that puts lower limits for the kernel weights. Hence, each kernel in the set of candidate kernels is used in the combined kernel and we obtain a more regularized learner. Their results on gene prioritization task coincide with that of Lewis et al. (2006a).

Bach et al. (2004) propose a modified primal formulation that uses weighted $l_1$-norm on feature spaces and $l_2$-norm within each feature space. The modified primal formulation is:

$$\begin{aligned}
&\text{minimize} && \frac{1}{2}\left(\sum_{m=1}^P \rho_m \left\|\boldsymbol{w}_m\right\|_2\right)^2 + C\sum_{i=1}^N \xi_i \\
&\text{with respect to} && \boldsymbol{w}_m \in \mathbb{R}^{D_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\
&\text{subject to} && y_i\left(\sum_{m=1}^P \left\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i)\right\rangle + b\right) \geq 1 - \xi_i \qquad \forall i
\end{aligned}$$

where the feature space $m$ has the dimensionality $D_m$ and the weight $\rho_m$. When we consider this optimization problem as a SOCP, we obtain the following dual formulation:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i \\
\text{with respect to} \quad & \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \gamma^2 \rho_m^2 \geq \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad \forall m \\
& \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned} \tag{5}$$

where we again get the optimal kernel weights from the optimal dual variables and the weights satisfy $\sum_{m=1}^{P} \rho_m^2 \eta_m = 1$. The dual problem is exactly equivalent to the QCQP formulation of Lanckriet et al. (2004b) when we take $\rho_m = \sqrt{\operatorname{tr}(\mathbf{K}_m)/c}$. The advantage of the SOCP formulation is that Bach et al. (2004) devise an SMO-like algorithm by adding a Moreau-Yosida regularization term, $1/2 \sum_{m=1}^{P} a_m^2 \|\boldsymbol{w}_m\|_2^2$, to the primal objective function and deriving the corresponding dual formulation. Using $l_1$-norm on feature spaces, Yamanishi et al. (2007) combine tree kernels for identifying human glycans into four blood components: leukemia cells, erythrocytes, plasma, and serum. Except on plasma task, representing glycans as rooted trees and combining kernels improve performance in terms of area under the curve. Özen et al. (2009) use the formulation of Bach et al. (2004) in order to combine different feature subsets for protein stability prediction problem and extract information about the importance of these subsets by looking at the learned kernel weights. Bach (2009) develops a method for learning linear combinations of an exponential number of kernels, which can be expressed as product of sums. The method is applied to nonlinear variable selection and efficiently explores the large feature spaces in polynomial time.

Sonnenburg et al. (2006a,b) rewrite the QCQP formulation of Bach et al. (2004):

$$\begin{aligned}
\text{minimize} \quad & \gamma \\
\text{with respect to} \quad & \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i \\
& \gamma \geq \underbrace{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{N} \alpha_i}_{S_m(\boldsymbol{\alpha})} \qquad \forall m
\end{aligned}$$



and convert this problem into the following SILP problem:

$$\begin{aligned}
\text{maximize} \quad & \theta \\
\text{with respect to} \quad & \theta \in \mathbb{R}, \boldsymbol{\eta} \in \mathbb{R}_+^P \\
\text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1 \\
& \sum_{m=1}^{P} \eta_m S_m(\boldsymbol{\alpha}) \geq \theta \qquad \forall \boldsymbol{\alpha} \in \left\{ \boldsymbol{\alpha} : \boldsymbol{\alpha} \in \mathbb{R}^N, \;\; \boldsymbol{\alpha}^T \boldsymbol{y} = 0, \;\; C \geq \boldsymbol{\alpha} \geq 0 \right\}
\end{aligned}$$

where the problem has infinitely many constraints due to possible values of $\boldsymbol{\alpha}$.

SILP formulation has lower computational complexity compared to SDP and QCQP formulations. Sonnenburg et al. (2006a,b) use a column generation approach to solve the resulting SILPs using a generic linear programming (LP) solver and a canonical SVM solver in the inner loop. Both the LP solver and the SVM solver can use previous optimal values for hot-start to obtain the new optimal values faster. These allow us to use SILP formulation to learn the kernel combination weights for hundreds of kernels on hundreds of thousands of training instances efficiently. For example, they perform training on a million real-world splice data set from computational biology with string kernels. They also generalize the idea to regression estimation, one-class classification, and strictly convex and differentiable loss functions.

Zien and Ong (2007) develop a QCQP formulation and convert this formulation in two different SILP problems for multiclass classification problems. They show that their formulation is the multiclass generalization of the previously developed binary classification methods of Bach et al. (2004) and Sonnenburg et al. (2006b). The proposed multiclass formulation is tested on different bioinformatics applications such as bacterial protein location prediction (Zien and Ong, 2007) and protein subcellular location prediction (Zien and Ong, 2007, 2008), and outperforms individual kernels and unweighted sum of kernels. Hu et al. (2009) combine the generalized multiple kernel learning formulation of Zien and Ong (2007) and the sparse kernel learning method of Wu et al. (2006). This hybrid approach learns the optimal kernel weights and also obtains a sparse solution.

Rakotomamonjy et al. (2007, 2008) propose a different primal problem for MKL and use a projected gradient method to solve this optimization problem. The proposed primal formulation is:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{P} \frac{1}{\eta_m} \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N} \xi_i \\
\text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{D_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \boldsymbol{\eta} \in \mathbb{R}_+^P \\
\text{subject to} \quad & y_i \left( \sum_{m=1}^{P} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i \\
& \sum_{m=1}^{P} \eta_m = 1
\end{aligned}$$

and they define the optimal SVM objective function value given $\boldsymbol{\eta}$ as $J(\boldsymbol{\eta})$:

$$\begin{aligned} \text{minimize} \;\; & J(\boldsymbol{\eta}) = \frac{1}{2}\sum_{m=1}^{P}\frac{1}{\eta_m}\|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{with respect to} \;\; & \boldsymbol{w}_m \in \mathbb{R}^{D_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \;\; & y_i\left(\sum_{m=1}^{P}\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i)\rangle + b\right) \geq 1 - \xi_i \qquad \forall i \; . \end{aligned}$$

Due to strong duality, one can also calculate $J(\boldsymbol{\eta})$ using the dual formulation:

$$\begin{aligned} \text{maximize} \;\; & J(\boldsymbol{\eta}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{with respect to} \;\; & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to} \;\; & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \; . \end{aligned}$$

The primal formulation can be seen as the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \;\; & J(\boldsymbol{\eta}) \\ \text{with respect to} \;\; & \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to} \;\; & \sum_{m=1}^{P}\eta_m = 1 \; . \end{aligned}$$

The overall procedure to solve this problem, called SIMPLEMKL, consists of two main steps: (a) solving a canonical SVM optimization problem with given $\boldsymbol{\eta}$ and (b) updating $\boldsymbol{\eta}$ using the following gradient calculated with $\boldsymbol{\alpha}$ found in the first step:

$$\frac{\partial J(\boldsymbol{\eta})}{\partial \eta_m} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad \forall m \; .$$

The gradient update procedure must consider the nonnegativity and normalization properties of the kernel weights. The derivative with respect to the kernel weights is exactly equivalent (up to a multiplicative constant) to the gradient of the margin calculated by Bousquet and Herrmann (2003). The overall algorithm is very similar to the algorithm used by Sonnenburg et al. (2006a,b) to solve SILP formulation. Both algorithms use a canonical SVM solver in order to calculate $\boldsymbol{\alpha}$ at each step. The difference is that they use different updating procedures for $\boldsymbol{\eta}$, namely, a projected gradient update and solving an LP. Rakotomamonjy et al. (2007, 2008) show that SIMPLEMKL is more stable than solving the SILP formulation. SIMPLEMKL can be generalized to regression estimation, one-class classification, and multiclass classification (Rakotomamonjy et al., 2008).



Xu et al. (2009a) propose a hybrid method that combines SILP formulation (Sonnenburg et al., 2006b) and SIMPLEMKL (Rakotomamonjy et al., 2008). SILP formulation does not regularize the kernel weights obtained from the cutting plane method and SIMPLEMKL uses the gradient calculated only in the last iteration. The proposed model overcomes both disadvantages and finds the kernel weights for the next iteration by solving a small QP problem; this regularizes the solution and uses the past information.

Longworth and Gales (2008, 2009) introduce an extra regularization term to the objective function of SIMPLEMKL (Rakotomamonjy et al., 2008). This modification allows us to change the level of sparsity for the kernels used in the combined kernel. The extra regularization term is:

$$\lambda\sum_{l=1}^{P}\left(\eta_m - \frac{1}{P}\right)^2 = \lambda\left(\sum_{l=1}^{P}\eta_m^2 - \frac{1}{P}\right) =^{+} \lambda\sum_{l=1}^{P}\eta_m^2$$

where $\lambda$ is regularization parameter that determines the solution sparsity. For example, large values of $\lambda$ force the mathematical model to use all kernels with uniform weight whereas small values produce sparse combinations.

Instead of selecting kernels from a predefined finite set, we can increase the number of candidate kernels in an iterative manner. Gehler and Nowozin (2008) propose a new algorithm that can basically select kernels from an uncountable infinite set. Their forward selection algorithm finds the kernel weights for a fixed size of candidate kernels by using one of the methods described above, then adds a new kernel to the set of candidate kernels, until convergence.

Most MKL methods do not consider the group structure between the kernels combined. For example, a group of kernels may be calculated on the same set of features and even if we assign a nonzero weight to only one of them, we have to extract the features in the testing phase. When kernels have such a group structure, it is reasonable to pick all or none of them in the combined kernel. Szafranski et al. (2008) follow this idea and derive a MKL method by changing the mathematical model used by Rakotomamonjy et al. (2007).

Subrahmanya and Shin (2009) generalize group-feature selection to kernel selection by introducing a log-based concave penalty term for obtaining extra sparsity; this is called sparse multiple kernel learning (SMKL). The reason for adding this concave penalty term is explained as the lack of ability of convex MKL methods to obtain sparse formulations. They show that SMKL obtains more sparse solutions than convex formulations for signal processing applications.

Tanabe et al. (2008) propose the following rule in order to choose the kernel weights for classification problems:

$$\eta_m = \frac{\pi_m - \delta}{\sum_{l=1}^{P}\pi_l - P\delta}$$

where $\pi_m$ is the accuracy obtained using only $\mathbf{K}_m$ and $\delta$ is the threshold that should be less than or equal to the minimum of the accuracies obtained from single kernel learners. Qiu and Lane (2009) propose two simple heuristics to select the kernel weights for regression

estimation problems:

$$\eta_m = \frac{R_m}{\sum_{l=1}^{P} R_l} \qquad \forall m$$

$$\eta_m = \frac{\sum_{l=1}^{P} M_l - M_m}{\sum_{k=1}^{P} \left( \sum_{l=1}^{P} M_l - M_k \right)} \qquad \forall m$$

where $R_m$ is the Pearson correlation coefficient the predicted labels generated by the regressor using kernel matrix $\mathbf{K}_m$ and the true labels, and $M_m$ is the mean square error generated by the regressor using kernel matrix $\mathbf{K}_m$.

#### 3.1.4 Binary Combination

Another possibility is to allow only binary $\eta_m$ for kernel selection. We get rid of kernels whose $\eta_m = 0$ and use the kernels whose $\eta_m = 1$.

Xu et al. (2009b) define a combined kernel over the set of kernels calculated on each feature independently and perform feature selection using this definition. The defined kernel function can be expressed as:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{D} \eta_m k(\boldsymbol{x}_i[m], \boldsymbol{x}_j[m])$$

where $\boldsymbol{\eta} \in \{0,1\}^D$. For efficient learning, $\boldsymbol{\eta}$ is relaxed into the continuous domain (i.e., $1 \geq \boldsymbol{\eta} \geq 0$). Following Lanckriet et al. (2004b), an SDP formulation is derived and this formulation is cast into a QCQP problem in order to reduce the time complexity.

### 3.2 Nonlinear Functions of Kernels

A linear combination may be restrictive and other combinations are also possible.

Ong et al. (2003) propose learning a kernel function instead of a kernel matrix. They define a kernel function in the space of kernels called a *hyperkernel*. Their construction includes convex combinations of an infinite number of pointwise nonnegative kernels. Ong and Smola (2003) and Ong et al. (2005) generalize hyperkernels to different machine learning problems such as classification, regression estimation, and one-class classification. When they use the regularized risk functional as the empirical quality functional to be optimized, the learning phase can be performed by solving an SDP problem. Tsang and Kwok (2006) convert the resulting optimization problems into second-order cone programming (SOCP) problems in order to reduce the time complexity of the training phase.

de Diego et al. (2004) define a functional form of combining two kernels:

$$\mathbf{K}_\eta = \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + f(\mathbf{K}_1 - \mathbf{K}_2)$$



where the term $f(\mathbf{K}_1 - \mathbf{K}_2)$ represents the difference of information between what $\mathbf{K}_1$ and $\mathbf{K}_2$ provide for classification. They investigate three different functions:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2}(k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + k_2(\boldsymbol{x}_i, \boldsymbol{x}_j)) + \tau y_i y_j |k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) - k_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|$$

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2}(k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + k_2(\boldsymbol{x}_i, \boldsymbol{x}_j)) + \tau y_i y_j (k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) - k_2(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$$

$$\mathbf{K}_\eta = \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + \tau(\mathbf{K}_1 - \mathbf{K}_2)(\mathbf{K}_1 - \mathbf{K}_2)$$

where $\tau \in \mathbb{R}_+$ is the parameter that represents the weight assigned to the term $f(\mathbf{K}_1 - \mathbf{K}_2)$ selected through cross-validation procedure and the first two functions do not ensure having positive semidefinite kernel matrices. More than two kernel functions can also be combined by applying these rules recursively.

Moguerza et al. (2004) propose a matrix functional form of combining kernels:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i, \boldsymbol{x}_j) k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

where $\eta_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ assigns a weight to $k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ according to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. They propose different heuristics to estimate the weighing functions.

Lee et al. (2007) combine kernels using a compositional method that constructs a $(P \times N) \times (P \times N)$ compositional kernel matrix. This matrix and $P$ times replicated training instances are used to train a canonical SVM. Lewis et al. (2006b) use a latent variable generative model using the maximum entropy discrimination to learn data-dependent kernel combination weights. This method combines a generative probabilistic model with a discriminative large margin method.

Gönen and Alpaydın (2008) combine kernels using weights calculated from a gating model. The proposed primal optimization problem is:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N} \xi_i \\ \text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{D_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \mathbf{V} \in \mathbb{R}^{P \times (D+1)} \\ \text{subject to} \quad & y_i \left( \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i|\mathbf{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i \end{aligned}$$

where the gating model parameterized by $\mathbf{V}$, $\eta_m(\boldsymbol{x}|\mathbf{V})$, assigns a weight to the feature space obtained with $\Phi_m(\boldsymbol{x})$. This optimization problem is not convex and a two-step alternate optimization procedure is used to find the classifier parameters and the gating model parameters. When we fix the gating model parameters, the problem becomes convex and we

obtain the following dual problem:

$$\begin{aligned} \text{maximize } & J(\mathbf{V}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_\eta \left( \boldsymbol{x}_i, \boldsymbol{x}_j \right) \\ \text{with respect to } & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to } & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i \end{aligned} \tag{6}$$

where the combined kernel matrix is represented as:

$$k_\eta \left( \boldsymbol{x}_i, \boldsymbol{x}_j \right) = \sum_{m=1}^{P} \eta_m \left( \boldsymbol{x}_i | \mathbf{V} \right) k_m \left( \boldsymbol{x}_i, \boldsymbol{x}_j \right) \eta_m \left( \boldsymbol{x}_j | \mathbf{V} \right) \ .$$

Assuming that the regions of expertise of kernels are linearly separable, we can express the gating model as:

$$\eta_m \left( \boldsymbol{x} | \mathbf{V} \right) = \frac{\exp \left( \langle \boldsymbol{v}_m, \Phi_\mathcal{G}(\boldsymbol{x}) \rangle + v_{m0} \right)}{\sum_{l=1}^{P} \exp \left( \langle \boldsymbol{v}_l, \Phi_\mathcal{G}(\boldsymbol{x}) \rangle + v_{l0} \right)} \quad \forall m$$

where $\mathbf{V} = \{ \boldsymbol{v}_1, v_{10}, \boldsymbol{v}_2, v_{20}, \ldots, \boldsymbol{v}_m, v_{m0} \}$, $\Phi_\mathcal{G}(\boldsymbol{x})$ is the mapping function for the gating feature space and there are $P(D_\mathcal{G}+1)$ parameters where $D_\mathcal{G}$ is the dimensionality of the gating feature space. The gating model parameters are updated at each iteration by calculating $\partial J(\mathbf{V}) / \partial \mathbf{V}$ and performing a gradient-descent step.

In some application areas such as bioinformatics, $\boldsymbol{x}$ vectors may appear in a nonvectorial format such as sequences, trees, and graphs. In such a case where we can calculate kernel values but can not represent the data instances as $\boldsymbol{x}$ vectors, directly, Gönen and Alpaydın (2009) define $\Phi_\mathcal{G}(\boldsymbol{x})$ in terms of the kernel values:

$$\Phi_\mathcal{G}(\boldsymbol{x}) = \left[ k_\mathcal{G}(\boldsymbol{x}_1, \boldsymbol{x}) \ \ k_\mathcal{G}(\boldsymbol{x}_2, \boldsymbol{x}) \ \ \ldots \ \ k_\mathcal{G}(\boldsymbol{x}_N, \boldsymbol{x}) \right]^T$$

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.

Varma and Babu (2007, 2009) propose a generalized formulation called generalized multiple kernel learning GMKL that contains two regularization terms and a loss function in the objective function. This formulation tries to regularize both the hyperplane weight and the kernel combination weights. The loss function can be one of the classical loss functions, such as hinge loss for classification or $\epsilon$-loss for regression estimation. The proposed primal formulation applied to binary classification problem with hinge loss and the regularization function $r(\boldsymbol{\eta})$ can be written as:

$$\begin{aligned} \text{minimize } & \frac{1}{2} \left\| \boldsymbol{w}_\eta \right\|_2^2 + C \sum_{i=1}^{N} \xi_i + r(\boldsymbol{\eta}) \\ \text{with respect to } & \boldsymbol{w}_\eta \in \mathbb{R}^{D_\eta}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to } & y_i \left( \langle \boldsymbol{w}_\eta, \Phi_\eta(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \quad \forall i \end{aligned}$$



This problem, different from the primal problem of SIMPLEMKL, is not convex, but the solution strategy is the same. The objective function value of the primal formulation given $\boldsymbol{\eta}$ is used as the target function:

$$\begin{aligned} \text{minimize } & J(\boldsymbol{\eta}) = \frac{1}{2} \left\| \boldsymbol{w}_\eta \right\|_2^2 + C \sum_{i=1}^{N} \xi_i + r(\boldsymbol{\eta}) \\ \text{with respect to } & \boldsymbol{w}_\eta \in \mathbb{R}^{D_\eta}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to } & y_i \left( \langle \boldsymbol{w}_\eta, \Phi_\eta(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \quad \forall i \end{aligned}$$

and the following dual formulation is used for the gradient step:

$$\begin{aligned} \text{maximize } & J(\boldsymbol{\eta}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_\eta \left( \boldsymbol{x}_i, \boldsymbol{x}_j \right) + r(\boldsymbol{\eta}) \\ \text{with respect to } & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to } & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i \ . \end{aligned}$$

The regularization function $r(\boldsymbol{\eta})$ and $k_\eta \left( \boldsymbol{x}_i, \boldsymbol{x}_j \right)$ can be any differentiable function of $\boldsymbol{\eta}$ with continuous derivative. The gradient with respect to the kernel weights is calculated as:

$$\frac{\partial J(\boldsymbol{\eta})}{\partial \eta_m} = \frac{\partial r(\boldsymbol{\eta})}{\partial \eta_m} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k_\eta \left( \boldsymbol{x}_i, \boldsymbol{x}_j \right)}{\partial \eta_m} \quad \forall m \ .$$

Varma and Babu (2007, 2009) perform gender identification experiments on a face image data set by combining kernels calculated on each individual feature, and hence, for kernels whose $\eta_m$ goes to 0 perform feature selection. The standard MKL and GMKL are trained with the kernel functions $k_\eta^S$ and $k_\eta^P$, respectively.

$$k_\eta^S = \sum_{m=1}^{D} \eta_m \exp \left( -\gamma_m \left( \boldsymbol{x}_i[m] - \boldsymbol{x}_j[m] \right)^2 \right)$$

$$k_\eta^P = \prod_{m=1}^{D} \exp \left( -\eta_m \left( \boldsymbol{x}_i[m] - \boldsymbol{x}_j[m] \right)^2 \right) = \exp \left( \sum_{m=1}^{D} -\eta_m \left( \boldsymbol{x}_i[m] - \boldsymbol{x}_j[m] \right)^2 \right)$$

where $[\cdot]$ indexes the elements of a vector. They show that GMKL with $k_\eta^P$ performs significantly better than the standard MKL with $k_\eta^S$. We see that using $k_\eta^P$ as the combined kernel function is equivalent to using different scaling parameters on each feature and using an RBF kernel over these scaled features with unit radius, as done by Grandvalet and Canu (2003).

Cortes et al. (2010) develop a nonlinear kernel combination method based on kernel ridge regression and polynomial combination of kernels. They propose to combine kernels

as follows:

$$k_\eta\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\sum_{\boldsymbol{q}\in Q}\eta_{q_1\ldots q_P}k_1\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)^{q_1}\ldots k_P\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)^{q_P}$$

where $Q=\{\boldsymbol{q}:\boldsymbol{q}\in\mathbb{R}_+^P,\ \sum_{m=1}^P q_m\leq d\}$ and $\eta_{q_1\ldots q_P}\geq 0$. The number of parameters to be learned is too large and the combined kernel is simplified in order to reduce the learning complexity:

$$k_\eta\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\sum_{\boldsymbol{q}\in R}\eta_1^{q_1}\ldots\eta_P^{q_P}k_1\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)^{q_1}\ldots k_P\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)^{q_P}$$

where $R=\{\boldsymbol{q}:\boldsymbol{q}\in\mathbb{R}_+^P,\ \sum_{m=1}^P q_m=d\}$ and $(q_1,\ldots,q_P)\in\mathbb{R}^P$. The combination weights are optimized using the following min-max optimization problem:

$$\min_{\boldsymbol{\eta}\in\mathcal{M}}\max_{\boldsymbol{\alpha}\in\mathbb{R}^N}-\boldsymbol{\alpha}^T\left(\mathbf{K}_\eta+\lambda\mathbf{I}\right)\boldsymbol{\alpha}+2\boldsymbol{\alpha}^T\boldsymbol{y}\ .$$

where $\mathcal{M}$ is a positive, bounded, and convex set. A projection-based gradient-descent algorithm can be utilized to solve this problem.

## 4. Combining Kernels using Kernel Similarity Measures

We can learn the kernel combination weights by using a quality measure that gives performance estimates for the kernel matrices calculated on training data. This corresponds to a function that assigns weights to kernel functions:

$$\eta_m=g_\eta(\mathbf{K}_1,\mathbf{K}_2,\ldots,\mathbf{K}_P)\ .$$

### 4.1 Combining Kernels using Kernel Alignment

Cristianini et al. (2002) define a notion of similarity between two kernels called *kernel alignment*. The empirical alignment of two kernels is calculated as follows:

$$\mathrm{A}(\mathbf{K}_1,\mathbf{K}_2)=\frac{\langle\mathbf{K}_1,\mathbf{K}_2\rangle_F}{\sqrt{\langle\mathbf{K}_1,\mathbf{K}_1\rangle_F\langle\mathbf{K}_2,\mathbf{K}_2\rangle_F}}$$

where $\langle\mathbf{K}_1,\mathbf{K}_2\rangle_F=\sum_{i=1}^N\sum_{j=1}^N k_1\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)k_2\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)$. This similarity measure can be seen as the cosine of the angle between $\mathbf{K}_1$ and $\mathbf{K}_2$. $\boldsymbol{y}\boldsymbol{y}^T$ can be defined as *ideal kernel* for a binary classification task, and the alignment between a kernel and the ideal kernel becomes:

$$\mathrm{A}(\mathbf{K},\boldsymbol{y}\boldsymbol{y}^T)=\frac{\left\langle\mathbf{K},\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F}{\sqrt{\langle\mathbf{K},\mathbf{K}\rangle_F\left\langle\boldsymbol{y}\boldsymbol{y}^T,\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F}}=\frac{\left\langle\mathbf{K},\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F}{N\sqrt{\langle\mathbf{K},\mathbf{K}\rangle_F}}\ .$$

Kernel alignment has one key property due to concentration (i.e., the probability of deviation from the mean is decayed exponentially), which enables us to keep the high alignment on a test set when we optimize it on a training set.



#### 4.1.1 Linear Combination

Lanckriet et al. (2004b) propose to optimize the kernel alignment as follows:

$$\begin{aligned}\text{maximize}\ &\ \mathrm{A}(\mathbf{K}_\eta^{\text{tra}},\boldsymbol{y}\boldsymbol{y}^T)\\ \text{with respect to}\ &\ \mathbf{K}_\eta\in\mathcal{K}_L\\ \text{subject to}\ &\ \mathrm{tr}\left(\mathbf{K}_\eta\right)=1\end{aligned}$$

where the trace of the combined kernel matrix is arbitrarily set to 1. This problem is equivalent to:

$$\begin{aligned}\text{maximize}\ &\ \left\langle\mathbf{K}_\eta^{\text{tra}},\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F\\ \text{with respect to}\ &\ \mathbf{K}_\eta\in\mathcal{K}_L\\ \text{subject to}\ &\ \left\langle\mathbf{K}_\eta,\mathbf{K}_\eta\right\rangle_F=1\\ &\ \mathrm{tr}\left(\mathbf{K}_\eta\right)=1\end{aligned}$$

and this problem can be converted into the following Sdp problem:

$$\begin{aligned}\text{maximize}\ &\ \left\langle\mathbf{K}_\eta^{\text{tra}},\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F\\ \text{with respect to}\ &\ \mathbf{K}_\eta\in\mathcal{K}_L,\mathbf{A}\in\mathbb{S}^N\\ \text{subject to}\ &\ \mathrm{tr}\left(\mathbf{A}\right)\leq 1\\ &\begin{pmatrix}\mathbf{A}&\mathbf{K}_\eta^T\\ \mathbf{K}_\eta&\mathbf{I}\end{pmatrix}\succeq 0\ .\end{aligned}$$

Igel et al. (2007) propose maximizing the kernel alignment using gradient-based optimization. They calculate the gradient of the alignment with respect to the kernel parameters:

$$\frac{\partial A(\mathbf{K}_\eta,\boldsymbol{y}\boldsymbol{y}^T)}{\partial\eta_m}=\frac{\left\langle\frac{\partial\mathbf{K}_\eta}{\partial\eta_m},\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F\langle\mathbf{K}_\eta,\mathbf{K}_\eta\rangle_F-\langle\mathbf{K}_\eta,\boldsymbol{y}\boldsymbol{y}^T\rangle_F\left\langle\frac{\partial\mathbf{K}_\eta}{\partial\eta_m},\mathbf{K}_\eta\right\rangle_F}{N\sqrt{\langle\mathbf{K}_\eta,\mathbf{K}_\eta\rangle_F^3}}\ .$$

In a transcription initiation site detection task for bacterial genes, they obtain better results by optimizing the kernel weights of the combined kernel function that is composed of six sequence kernels, using the gradient above.

#### 4.1.2 Nonnegative Linear Combination

(Lanckriet et al., 2004b) restrict the kernel weights to be nonnegative, their Sdp formulation reduces to the following Qcqp problem:

$$\begin{aligned}\text{maximize}\ &\ \sum_{m=1}^P\eta_m\left\langle\mathbf{K}_m^{\text{tra}},\boldsymbol{y}\boldsymbol{y}^T\right\rangle_F\\ \text{with respect to}\ &\ \boldsymbol{\eta}\in\mathbb{R}_+^P\\ \text{subject to}\ &\ \sum_{m=1}^P\sum_{l=1}^P\eta_m\eta_l\left\langle\mathbf{K}_m,\mathbf{K}_l\right\rangle_F\leq 1\ .\end{aligned}\tag{7}$$

Kandola et al. (2002) propose to maximize the alignment between a nonnegative linear combination of kernels and the ideal kernel. The alignment can be calculated as follows:

$$\mathrm{A}(\mathbf{K}_\eta, \boldsymbol{y}\boldsymbol{y}^T) = \frac{\sum_{m=1}^{P} \eta_m \left\langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F}{N\sqrt{\sum_{m=1}^{P}\sum_{l=1}^{P} \eta_m\eta_l \left\langle \mathbf{K}_m, \mathbf{K}_l \right\rangle_F}} .$$

We should choose kernel weights that maximize the alignment and this idea can be cast into the following optimization problem:

$$\begin{aligned} &\text{maximize} && \mathrm{A}(\mathbf{K}_\eta, \boldsymbol{y}\boldsymbol{y}^T) \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P \end{aligned}$$

and this problem is equivalent to:

$$\begin{aligned} &\text{maximize} && \sum_{m=1}^{P} \eta_m \left\langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P \\ &\text{subject to} && \sum_{m=1}^{P}\sum_{l=1}^{P} \eta_m\eta_l \left\langle \mathbf{K}_m, \mathbf{K}_l \right\rangle_F = c . \end{aligned}$$

Using the Lagrangian function, we can convert it into the following unconstrained optimization problem:

$$\begin{aligned} &\text{maximize} && \sum_{m=1}^{P} \eta_m \left\langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F - \mu \left( \sum_{m=1}^{P}\sum_{l=1}^{P} \eta_m\eta_l \left\langle \mathbf{K}_m, \mathbf{K}_l \right\rangle_F - c \right) \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P . \end{aligned}$$

Kandola et al. (2002) take $\mu = 1$ arbitrarily and add a regularization term to the objective function in order to prevent overfitting. The resulting QP is very similar to the hard margin SVM optimization problem and is expected to give sparse kernel combination weights.

$$\begin{aligned} &\text{maximize} && \sum_{m=1}^{P} \eta_m \left\langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F - \sum_{m=1}^{P}\sum_{l=1}^{P} \eta_m\eta_l \left\langle \mathbf{K}_m, \mathbf{K}_l \right\rangle_F - \lambda \sum_{m=1}^{P} \eta_m^2 \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P \end{aligned}$$

where we only learn the kernel combination weights.

#### 4.1.3 Convex Combination

Qiu and Lane (2009) propose the following simple heuristic to select the kernel weights using kernel alignment:

$$\eta_m = \frac{\mathrm{A}(\mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^T)}{\sum_{l=1}^{P} \mathrm{A}(\mathbf{K}_l, \boldsymbol{y}\boldsymbol{y}^T)} \qquad \forall m . \tag{8}$$



### 4.2 Combining Kernels using Other Similarity Measures

He et al. (2008) choose to optimize the distance between the combined kernel matrix and the ideal kernel, instead of optimizing the kernel alignment measure, by using the following optimization problem:

$$\begin{aligned} &\text{minimize} && \left\| \mathbf{K}_\eta - \boldsymbol{y}\boldsymbol{y}^T \right\|_F^2 \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P \\ &\text{subject to} && \sum_{m=1}^{P} \eta_m = 1 . \end{aligned}$$

This problem is equivalent to:

$$\begin{aligned} &\text{minimize} && \sum_{m=1}^{P}\sum_{l=1}^{P} \eta_m\eta_l \left\langle \mathbf{K}_m, \mathbf{K}_l \right\rangle_F - 2\sum_{m=1}^{P} \eta_m \left\langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P \\ &\text{subject to} && \sum_{m=1}^{P} \eta_m = 1 . \end{aligned} \tag{9}$$

Nguyen and Ho (2008) propose another quality measure called feature space-based kernel matrix evaluation measure (FSM) defined as:

$$\mathrm{FSM}(\mathbf{K}, \boldsymbol{y}) = \frac{s_+ + s_-}{\left\| \boldsymbol{m}_+ - \boldsymbol{m}_- \right\|_2}$$

where $\{s_+, s_-\}$ are standard deviations of the positive and negative classes, and $\{\boldsymbol{m}_+, \boldsymbol{m}_-\}$ are class centers in the feature space. Tanabe et al. (2008) optimize the kernel weights for the convex combination of kernels by minimizing this measure:

$$\begin{aligned} &\text{maximize} && \mathrm{FSM}(\mathbf{K}_\eta, \boldsymbol{y}) \\ &\text{with respect to} && \boldsymbol{\eta} \in \mathbb{R}_+^P \\ &\text{subject to} && \sum_{m=1}^{P} \eta_m = 1 . \end{aligned}$$

This method give similar performance results when compared to the SMO-like algorithm of Bach et al. (2004) for a protein-protein interaction prediction problem using much less time and memory.

Ying et al. (2009) follow an information theoretic approach based on the Kullback-Leibler (KL) divergence between the combined kernel matrix and the optimal kernel matrix:

$$\min_{\boldsymbol{\eta}} \mathrm{KL}\left( \mathcal{N}\left(\mathbf{0}, \mathbf{K}_\eta\right) || \mathcal{N}\left(\mathbf{0}, \boldsymbol{y}\boldsymbol{y}^T\right) \right) .$$

The kernel combinations weights can be optimized by using a projected gradient-descent method.

## 5. Combining Kernels using Boosting

Another possibility is to iteratively update by adding a new kernel to $f_\eta$ as training continues.

Inspired from ensemble and boosting methods, Bennett et al. (2002) modify the decision function in order to use multiple kernels:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \sum_{m=1}^{P} \alpha_{im} k_m(\boldsymbol{x}_i, \boldsymbol{x}) + b \,.$$

The model parameters, $\{\boldsymbol{\alpha}_{1:P}, b\}$, of kernel ridge regression model are learned using a gradient-descent algorithm in the function space. The columns of the combined kernel matrix are generated from the heterogeneous kernels on the fly. Bi et al. (2004) develop column generation boosting methods for binary classification and regression estimation problems. In each iteration, the proposed methods solve an LP or a QP on a working set depending on the regularization term used.

Crammer et al. (2003) modify the boosting methodology to work with kernels by rewriting two loss functions for a pair of data instances by considering the pair as a single instance:

$$\begin{aligned}
\text{ExpLoss}\left(k(\boldsymbol{x}_i, \boldsymbol{x}_j), y_i y_j\right) &= \exp\left(-y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \\
\text{LogLoss}\left(k(\boldsymbol{x}_i, \boldsymbol{x}_j), y_i y_j\right) &= \log\left(1 + \exp\left(-y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)\right) \,.
\end{aligned}$$

We iteratively update the combined kernel matrix by using one of these two loss functions.

## 6. Bayesian Kernel Combination

In a trained combiner parameterized by $\theta$, if we assume $\theta$ are random variables with a prior, we can use a Bayesian approach. For the case of a weighted sum, we can for example have a prior on the weights $\boldsymbol{\eta}$.

Girolami and Rogers (2005) formulate a Bayesian hierarchical model and derive variational Bayes estimators for classification and regression problems. The proposed decision function is:

$$\sum_{i=0}^{N} \alpha_i \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i, \boldsymbol{x})$$

where $\boldsymbol{\eta}$ is modeled with a Dirichlet prior and $\boldsymbol{\alpha}$ is modeled with a zero-mean Gaussian with an inverse gamma variance prior. Damoulas and Girolami (2009b) extend this method by adding auxiliary variables and developing a Gibbs sampler. Multinomial probit likelihood is used to obtain an efficient sampling procedure. Damoulas and Girolami (2008, 2009a) apply these methods to different bioinformatics problems, such as protein fold recognition and remote homology problems, and improve the prediction performances for these tasks.

Girolami and Zhong (2007) use the kernel combination idea for the covariance matrices in Gaussian processes. Instead of using a single covariance matrix, they define a weighted sum of covariance matrices calculated over different data sources. A joint inference is performed for both the Gaussian process coefficients and the kernel combination weights.



## 7. Experiments

We implement some representative MKL algorithms (see Table 2) in MATLAB and solve the optimization problems with MOSEK optimization software (Mosek, 2009). 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 $5 \times 2$ cross-validation to generate ten training and validation sets, with stratification. The validation sets of all folds are used to optimize hyperparameters (for example, trying values 0.001, 0.01, 0.1, 1, 10, 100, and 1000 for $C$). The best hyperparameter configuration (the one that has the highest average accuracy on the validation folds) is used to train the final learners on the training folds and their performance is measured over the test set. So, for each data set, we have ten validation set and ten test set results; we report their averages and one standard deviation error bars.

Table 2: Representative MKL algorithms used in the experiments.

| Algorithm | Explanation |
|---|---|
| MEAN | Solving (1) with $k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1/P) \sum_{m=1}^{P} k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ |
| PRODUCT | Solving (1) with $k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \prod_{m=1}^{P} k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ |
| CONIC | Solving (1) with $k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{\eta}$ from (7) |
| CONVEX | Solving (1) with $k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{\eta}$ from (9) |
| RATIO | Solving (1) with $k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{\eta}$ from (8) |
| MKL | Solving (5) |
| LMKL | Solving (6) in an iterative manner |

We perform simulations with three commonly used kernels: linear kernel ($k_L$), polynomial kernel ($k_P$), and Gaussian kernel ($k_G$). All kernel matrices are calculated and normalized to unit trace before training. We create a toy data set which are mixtures of Gaussians with different number of components (see Table 3) and we compare different MKL methods on this data set. We combine two different kernel sets in our experiments: (a) $k_L$ and $k_P$ with $q = 2, 3, 4, 5, 6, 7$ (b) $k_G$ with $s = 1/8, 1/4, 1/2, 1, 2, 4, 8$.

Table 3: Prior probabilities and Gaussian parameters used for toy data set GAUSS4 which has two components in each class.

| Data Set | Pos. Class ($priors, means, covariances$) | Neg. Class ($priors, means, covariances$) |
|---|---|---|
| GAUSS4 | $p_1 = 0.25\ \mu_1 = \begin{pmatrix} -3.0 \\ +1.0 \end{pmatrix}\ \Sigma_1 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$ | $p_3 = 0.25\ \mu_3 = \begin{pmatrix} -1.0 \\ -2.2 \end{pmatrix}\ \Sigma_3 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$ |
| | $p_2 = 0.25\ \mu_2 = \begin{pmatrix} +1.0 \\ +1.0 \end{pmatrix}\ \Sigma_2 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$ | $p_4 = 0.25\ \mu_4 = \begin{pmatrix} +3.0 \\ -2.2 \end{pmatrix}\ \Sigma_4 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$ |

Figure 1 compares the MKL methods with linear and polynomial kernels of different degrees on the GAUSS4 data set. We see that all methods obtain similar discriminant functions. MEAN, PRODUCT, and RATIO use all kernels in the combined kernel matrix,

whereas Conic, Convex, and Mkl use the linear and seventh degree polynomial kernels, and Lmkl uses only the seventh degree polynomial kernel.

Table 4 lists the performance results of single kernel Svms and the Mkl algorithms with linear and polynomial kernels of different degrees on the Gauss4 data set. There is no significant difference between accuracies obtained by single kernel Svms and the Mkl methods. However, the Mkl methods other than Mean and Product uses significantly fewer support vectors than Mean and Product. When we look at the training times, it is clear that multiple kernel methods require longer training times than single kernel Svms. Mkl and Lmkl takes much more time than other multiple kernel methods due to high computational complexity of optimization problem Mkl solves and iterative nature of Lmkl.

Table 4: Performances of single kernel Svms and representative Mkl algorithms with linear and polynomial kernels of different degrees on the Gauss4 data set.

| Kernel | Validation Acc. | Support Vector | Test Acc. | Tra. Time (sec.) |
|---|---|---|---|---|
| $k_L$ | $84.53 \pm 0.91$ | $44.13 \pm 1.51$ | $90.17 \pm 0.44$ | $1.39 \pm 0.06$ |
| $k_P$ $(q = 2)$ | $84.97 \pm 1.18$ | $36.92 \pm 2.16$ | $90.03 \pm 0.59$ | $1.40 \pm 0.04$ |
| $k_P$ $(q = 3)$ | $89.85 \pm 1.08$ | $24.68 \pm 0.78$ | $91.55 \pm 0.50$ | $1.70 \pm 0.02$ |
| $k_P$ $(q = 4)$ | $90.00 \pm 1.43$ | $22.75 \pm 0.67$ | $91.25 \pm 0.91$ | $1.77 \pm 0.20$ |
| $k_P$ $(q = 5)$ | $89.78 \pm 1.46$ | $24.00 \pm 0.51$ | $91.55 \pm 0.47$ | $1.78 \pm 0.20$ |
| $k_P$ $(q = 6)$ | $90.00 \pm 1.17$ | $24.73 \pm 2.17$ | $91.10 \pm 0.66$ | $1.97 \pm 0.26$ |
| $k_P$ $(q = 7)$ | $90.22 \pm 1.40$ | $25.18 \pm 2.03$ | $91.03 \pm 0.61$ | $2.12 \pm 0.16$ |
| Combination | Validation Acc. | Support Vector | Test Acc. | Tra. Time (sec.) |
| Mean | $90.05 \pm 1.31$ | $34.95 \pm 3.66$ | $91.08 \pm 0.58$ | $4.40 \pm 0.10$ |
| Product | $89.72 \pm 1.33$ | $34.90 \pm 0.65$ | $91.85 \pm 0.46$ | $4.59 \pm 0.08$ |
| Conic | $90.05 \pm 1.45$ | $27.02 \pm 2.59$ | $91.13 \pm 0.57$ | $6.00 \pm 0.24$ |
| Convex | $90.05 \pm 1.40$ | $24.85 \pm 1.81$ | $91.17 \pm 0.58$ | $5.92 \pm 0.29$ |
| Ratio | $89.70 \pm 1.17$ | $25.82 \pm 1.76$ | $91.47 \pm 0.84$ | $4.86 \pm 0.09$ |
| Mkl | $90.20 \pm 1.43$ | $25.73 \pm 2.28$ | $90.97 \pm 0.52$ | $15.16 \pm 1.25$ |
| Lmkl | $90.22 \pm 1.40$ | $25.45 \pm 1.96$ | $91.03 \pm 0.61$ | $26.32 \pm 2.05$ |

Figure 2 compares the Mkl methods with Gaussian kernels of different spreads on the Gauss4 data set. We clearly see that Product overfits, and Mean and Mkl store too many support vectors and tend to overfit. Lmkl uses only one kernel, whereas Conic and Convex use three kernels, and the other methods use more than three kernels.

Table 5 lists the performance results of single kernel Svms and the Mkl algorithms with Gaussian kernels of different spreads on the Gauss4 data set. There is no significant difference between accuracies obtained by single kernel Svms except $s = 1/8$ or 8 and the Mkl methods except Product. Single kernel Svms take much less time than multiple kernel methods. Fixed and alignment-based combination rules are faster than optimization-based approaches.



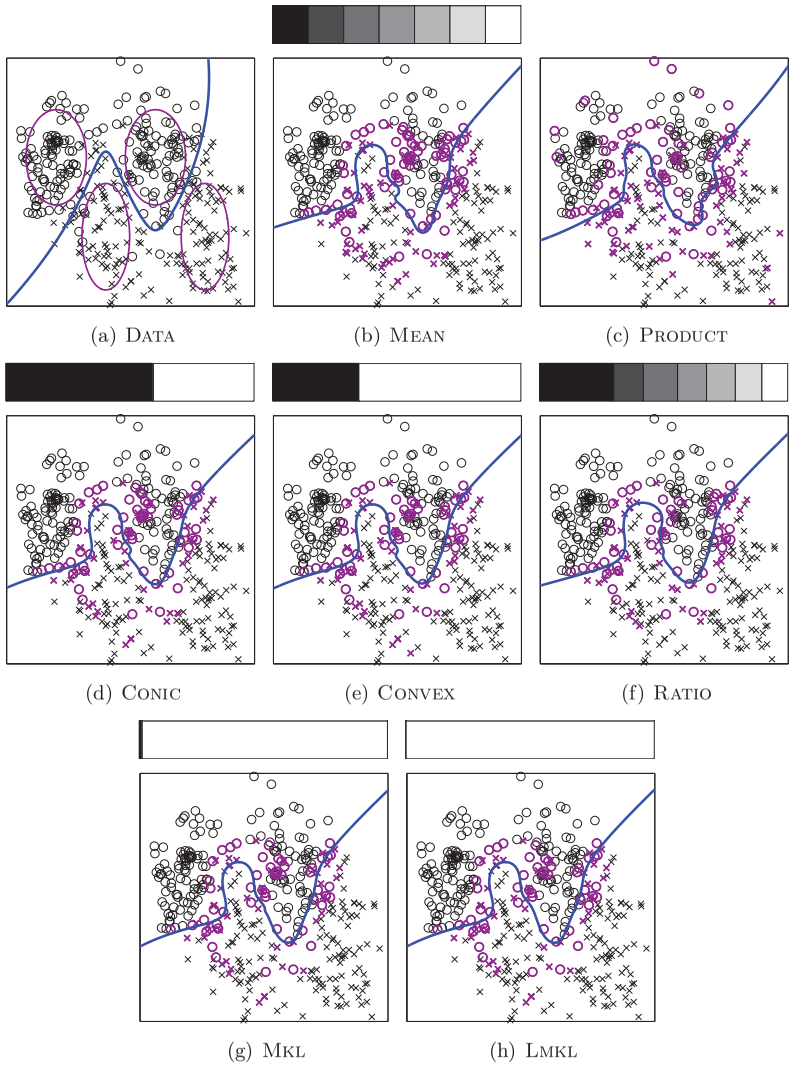
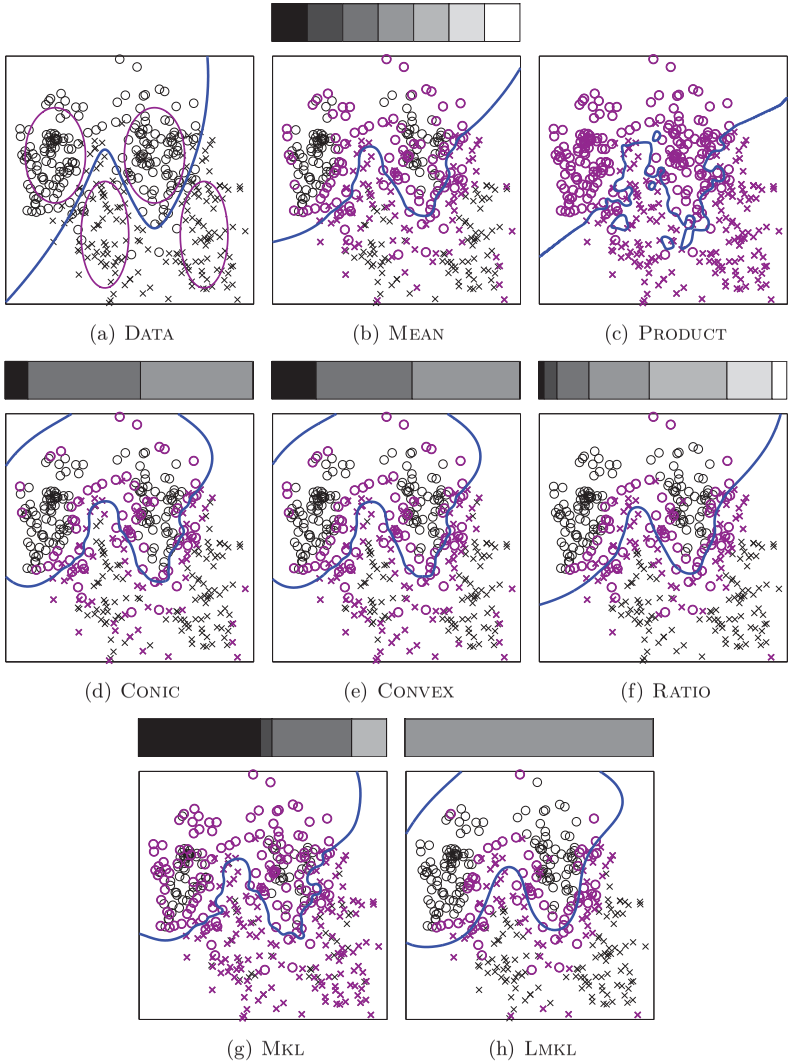



Figure 1: Representative Mkl algorithms with linear and polynomial kernels of different degrees on the Gauss4 data set. (a) Ellipses show the Gaussians from which data are sampled and the solid line shows the optimal Bayes' discriminant. (b)-(h) Learned discriminants (solid lines) and support vectors (bold points) on the Gauss4 data set. Bars on top of the figures represent the relative weights assigned to kernels.



(a) Data (b) Mean (c) Product

(d) Conic (e) Convex (f) Ratio

(g) Mkl (h) Lmkl

Figure 2: Comparison of representative Mkl algorithms with Gaussian kernels of different spreads on the Gauss4 data set. (a) Ellipses show the Gaussians from which data are sampled and the solid line shows the optimal Bayes' discriminant. (b)-(h) Learned discriminants (solid lines) and support vectors (bold points) on the Gauss4 data set. Bars on top of the figures represent the relative weights assigned to kernels.



Table 5: Performances of single kernel Svms and representative Mkl algorithms with Gaussian kernels of different spreads on the Gauss4 data set.

| Kernel | Validation Acc. | Support Vector | Test Acc. | Tra. Time (sec.) |
|---|---|---|---|---|
| $k_G$ $(s = 1/8)$ | $88.03 \pm 1.45$ | $100.00 \pm 0.00$ | $90.40 \pm 1.17$ | $0.64 \pm 0.01$ |
| $k_G$ $(s = 1/4)$ | $89.88 \pm 0.81$ | $100.00 \pm 0.00$ | $91.50 \pm 0.58$ | $1.25 \pm 0.03$ |
| $k_G$ $(s = 1/2)$ | $89.92 \pm 1.05$ | $100.00 \pm 0.00$ | $91.70 \pm 0.45$ | $2.17 \pm 0.25$ |
| $k_G$ $(s = 1)$ | $89.80 \pm 1.07$ | $26.13 \pm 1.77$ | $91.00 \pm 0.73$ | $2.13 \pm 0.02$ |
| $k_G$ $(s = 2)$ | $89.78 \pm 0.93$ | $37.50 \pm 2.76$ | $91.17 \pm 0.70$ | $2.08 \pm 0.02$ |
| $k_G$ $(s = 4)$ | $89.70 \pm 0.82$ | $44.20 \pm 2.32$ | $91.30 \pm 0.73$ | $1.96 \pm 0.02$ |
| $k_G$ $(s = 8)$ | $86.33 \pm 1.08$ | $57.30 \pm 2.32$ | $89.53 \pm 0.88$ | $1.84 \pm 0.02$ |
| Combination | Validation Acc. | Support Vector | Test Acc. | Tra. Time (sec. |
| Mean | $89.28 \pm 0.99$ | $50.27 \pm 2.25$ | $91.75 \pm 0.67$ | $3.43 \pm 0.03$ |
| Product | $87.45 \pm 1.21$ | $100.00 \pm 0.00$ | $90.00 \pm 1.12$ | $2.08 \pm 0.01$ |
| Conic | $89.72 \pm 0.98$ | $35.67 \pm 5.95$ | $91.50 \pm 0.61$ | $5.00 \pm 0.05$ |
| Convex | $89.47 \pm 1.02$ | $43.40 \pm 3.56$ | $91.85 \pm 0.54$ | $4.99 \pm 0.04$ |
| Ratio | $89.60 \pm 0.89$ | $34.77 \pm 1.93$ | $91.83 \pm 0.58$ | $3.67 \pm 0.02$ |
| Mkl | $89.30 \pm 0.90$ | $77.22 \pm 2.95$ | $91.40 \pm 0.46$ | $12.92 \pm 0.13$ |
| Lmkl | $89.65 \pm 1.05$ | $33.00 \pm 2.05$ | $91.60 \pm 0.56$ | $25.54 \pm 1.76$ |

We also perform experiments on the Multi-Feature digit recognition data set from the UCI repository, composed of six different data representations for 2000 handwritten numerals. Two binary classification problems are generated from the Multi-Feature data set: In the Eo data set, we separate even digits from odd digits; in the Sl data set, we separate small ('0' - '4') digits from large ('5' - '9') digits. The properties of these different data representations are summarized in Table 6. In our experiments, we combine six linear kernels calculated from each of the representations by using different Mkl methods.

Table 6: Different sources in the Multi-Feature data set.

| Source | Dimension | Explanation |
|---|---|---|
| Fac | 216 | Profile correlations |
| Fou | 76 | Fourier coefficients of the character shapes |
| Kar | 64 | Karhunen-Loève coefficients |
| Mor | 6 | Morphological features |
| Pix | 240 | Pixel averages in $2 \times 3$ windows |
| Zer | 47 | Zernike moments |

Table 7 lists the performance results of single kernel Svms and the Mkl algorithms on the Multi-Feature Eo data set. There is no significant difference between the best single kernel Svm accuracy obtained by using Fac and accuracies of the Mkl methods except Product.

Table 7: Performances of single kernel SVMs and representative MKL algorithms on the MULTI-FEATURE EO data set.

| Source | Validation Acc. | Support Vector | Test Acc. |
|---|---|---|---|
| FAC | $97.79 \pm 0.50$ | $10.30 \pm 0.83$ | $98.39 \pm 0.36$ |
| FOU | $80.63 \pm 1.08$ | $52.92 \pm 0.89$ | $79.69 \pm 1.11$ |
| KAR | $94.74 \pm 0.64$ | $26.87 \pm 0.72$ | $95.40 \pm 0.36$ |
| MOR | $96.10 \pm 0.50$ | $21.37 \pm 0.81$ | $95.96 \pm 0.50$ |
| PIX | $70.07 \pm 0.31$ | $71.34 \pm 0.78$ | $70.15 \pm 0.00$ |
| ZER | $82.44 \pm 1.40$ | $38.77 \pm 1.35$ | $80.63 \pm 0.71$ |
| Combination | Validation Acc. | Support Vector | Test Acc. |
| MEAN | $98.57 \pm 0.22$ | $15.67 \pm 0.71$ | $98.21 \pm 0.35$ |
| PRODUCT | $96.36 \pm 0.60$ | $100.00 \pm 0.00$ | $95.87 \pm 0.31$ |
| CONIC | $98.33 \pm 0.38$ | $15.75 \pm 0.65$ | $98.31 \pm 0.25$ |
| CONVEX | $97.79 \pm 0.50$ | $10.30 \pm 0.83$ | $98.39 \pm 0.36$ |
| RATIO | $98.66 \pm 0.19$ | $14.36 \pm 0.75$ | $98.22 \pm 0.28$ |
| MKL | $98.36 \pm 0.36$ | $14.88 \pm 0.81$ | $98.31 \pm 0.34$ |
| LMKL | $98.69 \pm 0.34$ | $11.23 \pm 1.58$ | $98.15 \pm 0.27$ |

Table 8 lists the performance results of single kernel SVMs and the MKL algorithms on the MULTI-FEATURE SL data set. All MKL methods achieve better accuracy results than the best single kernel SVM accuracy obtained by using FAC.

## 8. Conclusions

There is a significant amount of work on multiple kernel learning methods. This is because in many applications, one can come up with many possible kernel functions and instead of choosing one among them, we are interested in an algorithm that can automatically determine which ones are useful, which ones are not useful and therefore can be pruned, and best combine the useful ones. Or, in some applications we may have different sources of information coming from different modalities or corresponding to results from different experimental methodologies each having its own (or possibly multiple) kernel(s). In such a case, a good procedure for kernel combination means a good combination of inputs from those multiple sources.

In this paper, we give a taxonomy of multiple kernel learning algorithms to best highlight the similarities and differences among the proposed algorithms in the literature, which we then review in detail.

We also perform experiments on synthetic and real data comparing several multiple kernel methods in practice. We see that though there may not be such a drastic difference in terms of accuracy between them, different multiple kernel methods differ in the complexity of their solution, as given by the number of kernel selected and the number of stored support



Table 8: Performances of single kernel SVMs and representative MKL algorithms on the MULTI-FEATURE SL data set.

| Source | Validation Acc. | Support Vector | Test Acc. |
|---|---|---|---|
| FAC | $93.58 \pm 0.49$ | $17.93 \pm 0.91$ | $94.97 \pm 0.87$ |
| FOU | $90.13 \pm 1.16$ | $28.90 \pm 1.69$ | $90.54 \pm 1.12$ |
| KAR | $85.80 \pm 0.76$ | $33.62 \pm 1.31$ | $88.13 \pm 0.73$ |
| MOR | $88.29 \pm 1.04$ | $46.35 \pm 1.64$ | $89.42 \pm 0.65$ |
| PIX | $69.49 \pm 0.89$ | $67.93 \pm 12.91$ | $69.91 \pm 1.26$ |
| ZER | $90.39 \pm 0.62$ | $26.27 \pm 1.67$ | $89.12 \pm 0.63$ |
| Combination | Validation Acc. | Support Vector | Test Acc. |
| MEAN | $97.43 \pm 0.34$ | $21.67 \pm 0.50$ | $97.78 \pm 0.31$ |
| PRODUCT | $96.17 \pm 0.40$ | $97.34 \pm 0.51$ | $95.93 \pm 0.17$ |
| CONIC | $96.72 \pm 0.38$ | $23.46 \pm 0.87$ | $96.93 \pm 0.27$ |
| CONVEX | $96.48 \pm 0.59$ | $33.78 \pm 0.90$ | $96.46 \pm 0.34$ |
| RATIO | $97.32 \pm 0.33$ | $20.89 \pm 0.89$ | $97.78 \pm 0.28$ |
| MKL | $97.13 \pm 0.43$ | $32.59 \pm 0.82$ | $97.40 \pm 0.37$ |
| LMKL | $97.39 \pm 0.81$ | $14.35 \pm 1.43$ | $97.73 \pm 0.57$ |

vectors, and because they correspond to solutions of optimization problems complexity, in the training time.

## 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

Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Prooceeding of the 18th Conference on Learning Theory*, 2005.

Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Pontil. A DC-programming algorithm for kernel selection. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Francis R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems 21*, 2009.

Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl 1):i38–46, 2005.

Kristin P. Bennett, Michinari Momma, and Mark J. Embrechts. MARK: A boosting algorithm for heterogeneous kernel models. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

Jinbo Bi, Tong Zhang, and Kristin P. Bennett. Column-generation boosting methods for mixture of kernels. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

Olivier Bousquet and Daniel J. L. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems 15*, 2003.

Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

Domenico Conforti and Rosita Guido. Kernel based support vector machine via semidefinite programming: Application to medical diagnosis. *Computers and Operations Research*, 2009.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. $L_2$ regularization for learning kernels. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems 22*, 2010.

Koby Crammer, Joseph Keshet, and Yoram Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems 15*, 2003.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

Nello Cristianini, John Shawe-Taylor, Andree Elisseef, and Jaz Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, 2002.

Theodoros Damoulas and Mark A. Girolami. Probabilistic multi-class multi-kernel learning: On protein fold recognition and remote homology detection. *Bioinformatics*, 24(10):1264–1270, 2008.

Theodoros Damoulas and Mark A. Girolami. Combining feature spaces for classification. *Pattern Recognition*, 42(11):2671–2683, 2009a.

Theodoros Damoulas and Mark A. Girolami. Pattern recognition with a bayesian kernel combination machine. *Pattern Recognition Letters*, 30(1):46–54, 2009b.



Tijl De Bie, Leon-Charles Tranchevent, Liesbeth M. M. van Oeffelen, and Yves Moreau. Kernel-based data fusion for gene prioritization. *Bioinformatics*, 23(13):i125–132, 2007.

Isaac Martín de Diego, Javier M. Moguerza, and Alberto Muñoz. Combining kernel information for support vector classification. In *Proceedings of the 4th International Workshop Multiple Classifier Systems*, 2004.

Réda Dehak, Najim Dehak, Patrick Kenny, and Pierre Dumouchel. Kernel combination for SVM speaker verification. In *Proceedings of the Speaker and Language Recognition Workshop*, 2008.

Glenn Fung, Murat Dundar, Jinbo Bi, and Bharat Rao. A fast iterative algorithm for Fisher discriminant using heterogeneous kernels. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Peter Vincent Gehler and Sebastian Nowozin. Infinite kernel learning. Technical report, Max Planck Institute for Biological Cybernetics, 2008.

Mark Girolami and Simon Rogers. Hierarchic bayesian models for kernel learning. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Mark Girolami and Mingjun Zhong. Data integration for classification problems employing Gaussian process priors. In *Advances in Neural Processing Systems 19*, 2007.

Mehmet Gönen and Ethem Alpaydın. Localized multiple kernel learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel machines using localized kernels. In *Supplementary Proceedings of the 4th IAPR International Conference on Pattern Recognition in Bioinformatics*, 2009.

Yves Grandvalet and Stéphane Canu. Adaptive scaling for feature selection in SVMs. In *Advances in Neural Information Processing Systems 15*, 2003.

Junfeng He, Shih-Fu Chang, and Lexing Xie. Fast kernel learning for spatial pyramid matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Mingqing Hu, Yiqiang Chen, and James Tin-Yau Kwok. Building sparse multiple-kernel SVM classifiers. *IEEE Transactions on Neural Networks*, 20(5):827–839, 2009.

Christian Igel, Tobias Glasmachers, Britta Mersch, Nico Pfeifer, and Peter Meinicke. Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):216–226, 2007.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. Optimizing kernel alignment over combinations of kernels. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

Seung-Jean Kim, Alessandro Magnani, and Stephen Boyd. Optimal kernel selection in kernel Fisher discriminant analysis. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate $l_p$-norm multiple kernel learning. In *Advances in Neural Information Processing Systems 22*, 2010.

G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing*, 2004a.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004b.

Gert R. G. Lanckriet, Tijl de Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004c.

Wan-Jui Lee, Sergey Verzakov, and Robert P. W. Duin. Kernel combination versus classifier combination. In *Proceedings of the 7th International Workshop Multiple Classifier Systems*, 2007.

Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Support vector machine learning from heterogeneous data: An empirical analysis using protein sequence and structure. *Bioinformatics*, 22(22):2753–2760, 2006a.

Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Nonstationary kernel combination. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006b.

Yen-Yu Lin, Tyng-Luh Liu, and Chiou-Shann Fuh. Dimensionality reduction for data in multiple feature representations. In *Advances in Neural Processing Systems 21*, 2009.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

C. Longworth and M. J. F. Gales. Multiple kernel learning for speaker verification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.



C. Longworth and M. J. F. Gales. Combining derivative and parametric kernels for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):748–757, 2009.

Brian McFee and Gert Lanckriet. Partial order embedding with multiple kernels. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

Javier M. Moguerza, Alberto Muñoz, and Isaac Martín de Diego. Improving support vector classification via the combination of multiple sources of information. In *Proceedings of the Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops*, 2004.

Mosek. *The MOSEK Optimization Tools Manual Version 6.0 (Revision 55)*. MOSEK ApS, Denmark, 2009.

Canh Hao Nguyen and Tu Bao Ho. An efficient kernel matrix evaluation measure. *Pattern Recognition*, 41(11):3366–3372, 2008.

Cheng Soon Ong and Alexander J. Smola. Machine learning using hyperkernels. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Hyperkernels. In *Advances in Neural Information Processing Systems 15*, 2003.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

Ayşegül Özen, Mehmet Gönen, Ethem Alpaydın, and Türkan Haliloğlu. Machine learning integration for predicting the effect of single amino acid substitutions on protein stability. *BMC Structural Biology*, 9(1):66, 2009.

Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the 5th Annual International Conference on Computational Molecular Biology*, 2001.

Shibin Qiu and Terran Lane. Multiple kernel learning for support vector regression. Technical report, Computer Science Department, University of New Mexico, 2005.

Shibin Qiu and Terran Lane. A framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(2):190–199, 2009.

Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors. *Kernel Methods in Computational Biology*. The MIT Press, 2004.

Sören Sonnenburg, Gunnar Rätsch, and Christin Schäfer. A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processing Systems 18*, 2006a.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006b.

Niranjan Subrahmanya and Yung C. Shin. Sparse multiple kernel learning for signal processing applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Ying Tan and Jun Wang. A support vector machine with a hybrid kernel and minimal Vapnik-Chervonenkis dimension. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):385–395, 2004.

Hiroaki Tanabe, Tu Bao Ho, Canh Hao Nguyen, and Saori Kawasaki. Simple but effective methods for combining kernels in computational biology. In *Proceedings of IEEE International Conference on Research, Innovation and Vision for the Future*, 2008.

Ivor Wai-Hung Tsang and James Tin-Yau Kwok. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17(1):48–58, 2006.

Koji Tsuda, Shinsuke Uda, Taishin Kin, and Kiyoshi Asai. Minimizing the cross validation error to mix kernel matrices of heterogeneous biological data. *Neural Processing Letters*, 19(1):63–72, 2004.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. John Wiley & Sons, 1998.

Manik Varma and Bodla Rakesh Babu. Learning the discriminative power-invariance trade-off. In *Proceedings of the International Conference in Computer Vision*, 2007.

Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems 13*, 2001.

Mingrui Wu, Bernhard Schölkopf, and Gökhan Bakır. A direct method for building sparse kernel learning algorithms. *Journal of Machine Learning Research*, 7:603–624, 2006.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Processing Systems 17*, 2005.



Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems 21*, 2009a.

Zenglin Xu, Rong Jin, Jieping Ye, Michael R. Lyu, and Irwin King. Non-monotonic feature selection. In *Proceedings of the 26th International Conference on Machine Learning*, 2009b.

Yoshihiro Yamanishi, Francis Bach, and Jean-Philippe Vert. Glycan classification with tree kernels. *Bioinformatics*, 23(10):1211–1216, 2007.

Fei Yan, Krystian Mikolajczyk, Josef Kittler, and Muhammad Tahir. A comparison of $l_1$ norm and $l_2$ norm multiple kernel SVMs in image and video classification. In *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing*, pages 7–12, 2009.

Jieping Ye, Jianhui Chen, and Shuiwang Ji. Discriminant kernel and regularization parameter learning via semidefinite programming. In *Proceedings of the 24th International Conference on Machine Learning*, 2007a.

Jieping Ye, Shuiwang Ji, and Jianhui Chen. Learning the kernel matrix in discriminant analysis via quadratically constrained quadratic programming. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007b.

Jieping Ye, Shuiwang Ji, and Jianhui Chen. Multi-class discriminant kernel learning via convex programming. *Journal of Machine Learning Research*, 9:719–758, 2008.

Yiming Ying, Kaizhu Huang, and Colin Campbell. Enhanced protein fold recognition through a novel data integration approach. *BMC Bioinformatics*, 10(1):267, 2009.

Bin Zhao, James T. Kwok, and Changshui Zhang. Multiple kernel clustering. In *Proceedings of the 9th SIAM International Conference on Data Mining*, 2009.

Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Alexander Zien and Cheng Soon Ong. An automated combination of kernels for predicting protein subcellular localization. In *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics*, 2008.