Journal of Machine Learning Research 12 (2011) 2211-2268 Submitted 12/09; Revised 9/10; Published 7/11

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

Editor: Francis Bach

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 learning algorithms. We perform experiments on real data sets for better illustration and comparison of existing algorithms. 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 complexity. We see that overall, using multiple kernels instead of a single one is useful and believe that combining kernels in a nonlinear or data-dependent way seems more promising than linear combination in fusing information provided by simple linear kernels, whereas linear methods are more reasonable when combining complex Gaussian kernels.

Keywords: support vector machines, kernel machines, multiple kernel learning

1. Introduction

The 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 $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i$ 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 $\Phi\colon \mathbb{R}^D \to \mathbb{R}^S$. The resulting discriminant function is

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

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

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

where $\mathbf{w}$ is the vector of weight coefficients, C is a predefined positive trade-off parameter between model simplicity and classification error, ξ 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:

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

where $k\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is named the *kernel function* and α is the vector of dual variables corresponding to each separation constraint. Solving this, we get $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$ and the discriminant function can be rewritten as

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

There are several kernel functions successfully used in the literature, such as the linear kernel (k_{LIN}), the polynomial kernel (k_{POL}), and the Gaussian kernel (k_{GAU}):

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

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

Selecting the kernel function $k(\cdot,\cdot)$ 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:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = f_\eta(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\}_{m=1}^P)$$

where the combination function, $f_\eta \colon \mathbb{R}^P \to \mathbb{R}$, can be a linear or a nonlinear function. Kernel functions, $\{k_m\colon \mathbb{R}^{D_m} \times \mathbb{R}^{D_m} \to \mathbb{R}\}_{m=1}^P$, take P feature representations (not necessarily different) of data instances: $\mathbf{x}_i = \{\mathbf{x}_i^m\}_{m=1}^P$ where $\mathbf{x}_i^m \in \mathbb{R}^{D_m}$, and D_m is the dimensionality of the corresponding feature representation. η parameterizes the combination function and the more common implementation is

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = f_\eta(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\}_{m=1}^P | \eta)$$

where the parameters are used to combine a set of predefined kernels (i.e., we know the kernel functions and corresponding kernel parameters before training). It is also possible to view this as

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = f_\eta(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m | \eta)\}_{m=1}^P)$$

where the parameters integrated into the kernel functions are optimized during training. Most of the existing MKL algorithms fall into the first category and try to combine predefined kernels in an optimal way. We will discuss the algorithms in terms of the first formulation but give the details of the algorithms that use the second formulation where appropriate.

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 MKL: (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 inputs coming from different representations possibly from different sources or modalities. Since these are different representations, they have different measures of similarity corresponding to different kernels. In such a case, combining kernels is one possible way to combine multiple information sources. Noble (2004) calls 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 is significant amount of work in the literature for combining multiple kernels. Section 2 identifies the key properties of the existing MKL algorithms in order to construct a taxonomy, highlighting similarities and differences between them. Section 3 categorizes and discusses the existing MKL algorithms with respect to this taxonomy. We give experimental results in Section 4 and conclude in Section 5. The lists of acronyms and notation used in this paper are given in Appendices A and B, respectively.

2. Key Properties of Multiple Kernel Learning

We identify and explain six key properties of the existing MKL algorithms in order to obtain a meaningful categorization. We can think of these six dimensions (though not necessarily orthogonal) defining a space in which we can situate the existing MKL algorithms and search for structure (i.e., groups) to better see the similarities and differences between them. These properties are the learning method, the functional form, the target function, the training method, the base learner, and the computational complexity.

2.1 The Learning Method

The existing MKL algorithms use different learning methods for determining the kernel combination function. We basically divide them into five major categories:

1. *Fixed rules* are functions without any parameters (e.g., summation or multiplication of the kernels) and do not need any training.

2. *Heuristic approaches* use a parameterized combination function and find the parameters of this function generally by looking at some measure obtained from each kernel function separately. These measures can be calculated from the kernel matrices or taken as the performance values of the single kernel-based learners trained separately using each kernel.

3. *Optimization approaches* also use a parametrized combination function and learn the parameters by solving an optimization problem. This optimization can be integrated to a kernel-based learner or formulated as a different mathematical model for obtaining only the combination parameters.

4. *Bayesian approaches* interpret the kernel combination parameters as random variables, put priors on these parameters, and perform inference for learning them and the base learner parameters.

5. *Boosting approaches*, inspired from ensemble and boosting methods, iteratively add a new kernel until the performance stops improving.

2.2 The Functional Form

There are different ways in which the combination can be done and each has its own combination parameter characteristics. We group functional forms of the existing MKL algorithms into three basic categories:

1. *Linear combination* methods are the most popular and have two basic categories: unweighted sum (i.e., using sum or mean of the kernels as the combined kernel) and weighted sum. In the weighted sum case, we can linearly parameterize the combination function:

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

where η denotes the kernel weights. Different versions of this approach differ in the way they put restrictions on η: the linear sum (i.e., $\eta \in \mathbb{R}^P$), the conic sum (i.e., $\eta \in \mathbb{R}_+^P$), or the convex sum (i.e., $\eta \in \mathbb{R}_+^P$ and $\sum_{m=1}^P \eta_m = 1$). As can be seen, the conic sum is a special case of the linear sum and the convex sum is a special case of the conic sum. The conic and convex sums have two advantages over the linear sum in terms of interpretability. First, when we have positive kernel weights, we can extract the relative importance of the combined kernels by looking at them. Second, when we restrict the kernel weights to be nonnegative, this corresponds to scaling the feature spaces and using the concatenation of them as the combined feature representation:

$$\Phi_\eta(\mathbf{x}) = \begin{pmatrix} \sqrt{\eta_1}\Phi_1(\mathbf{x}^1) \\ \sqrt{\eta_2}\Phi_2(\mathbf{x}^2) \\ \vdots \\ \sqrt{\eta_P}\Phi_P(\mathbf{x}^P) \end{pmatrix}$$

and the dot product in the combined feature space gives the combined kernel:

$$\langle \Phi_\eta(\mathbf{x}_i), \Phi_\eta(\mathbf{x}_j) \rangle = \begin{pmatrix} \sqrt{\eta_1}\Phi_1(\mathbf{x}_i^1) \\ \sqrt{\eta_2}\Phi_2(\mathbf{x}_i^2) \\ \vdots \\ \sqrt{\eta_P}\Phi_P(\mathbf{x}_i^P) \end{pmatrix}^\top \begin{pmatrix} \sqrt{\eta_1}\Phi_1(\mathbf{x}_j^1) \\ \sqrt{\eta_2}\Phi_2(\mathbf{x}_j^2) \\ \vdots \\ \sqrt{\eta_P}\Phi_P(\mathbf{x}_j^P) \end{pmatrix} = \sum_{m=1}^P \eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m).$$

The combination parameters can also be restricted using extra constraints, such as the ℓ_p-norm on the kernel weights or trace restriction on the combined kernel matrix, in addition to their domain definitions. For example, the ℓ_1-norm promotes sparsity on the kernel level, which can be interpreted as feature selection when the kernels use different feature subsets.

2. *Nonlinear combination* methods use nonlinear functions of kernels, namely, multiplication, power, and exponentiation.

3. *Data-dependent combination* methods assign specific kernel weights for each data instance. By doing this, they can identify local distributions in the data and learn proper kernel combination rules for each region.

2.3 The Target Function

We can optimize different target functions when selecting the combination function parameters. We group the existing target functions into three basic categories:

1. *Similarity-based functions* calculate a similarity metric between the combined kernel matrix and an optimum kernel matrix calculated from the training data and select the combination function parameters that maximize the similarity. The similarity between two kernel matrices can be calculated using kernel alignment, Euclidean distance, Kullback-Leibler (KL) divergence, or any other similarity measure.

2. *Structural risk functions* follow the structural risk minimization framework and try to minimize the sum of a regularization term that corresponds to the model complexity and an error term that corresponds to the system performance. The restrictions on kernel weights can be integrated into the regularization term. For example, structural risk function can use the ℓ_1-norm, the ℓ_2-norm, or a mixed-norm on the kernel weights or feature spaces to pick the model parameters.

3. *Bayesian functions* measure the quality of the resulting kernel function constructed from candidate kernels using a Bayesian formulation. We generally use the likelihood or the posterior as the target function and find the maximum likelihood estimate or the maximum a posteriori estimate to select the model parameters.

2.4 The Training Method

We can divide the existing MKL algorithms into two main groups in terms of their training methodology:

1. *One-step methods* calculate both the combination function parameters and the parameters of the combined base learner in a single pass. One can use a sequential approach or a simultaneous approach. In the sequential approach, the combination function parameters are determined first, and then a kernel-based learner is trained using the combined kernel. In the simultaneous approach, both set of parameters are learned together.

2. *Two-step methods* use an iterative approach where each iteration, first we update the combination function parameters while fixing the base learner parameters, and then we update the base learner parameters while fixing the combination function parameters. These two steps are repeated until convergence.

2.5 The Base Learner

There are many kernel-based learning algorithms proposed in the literature and all of them can be transformed into an MKL algorithm, in one way or another.

The most commonly used base learners are SVM and support vector regression (SVR), due to their empirical success, their ease of applicability as a building block in two-step methods, and their ease of transformation to other optimization problems as a one-step training method using the simultaneous approach. Kernel Fisher discriminant analysis (KFDA), regularized kernel discriminant analysis (RKDA), and kernel ridge regression (KRR) are three other popular methods used in MKL.

Multinomial probit and Gaussian process (GP) are generally used in Bayesian approaches. New inference algorithms are developed for modified probabilistic models in order to learn both the combination function parameters and the base learner parameters.

2.6 The Computational Complexity

The computational complexity of an MKL algorithm mainly depends on its training method (i.e., whether it is one-step or two-step) and the computational complexity of its base learner.

One-step methods using fixed rules and heuristics generally do not spend much time to find the combination function parameters, and the overall complexity is determined by the complexity of the base learner to a large extent. One-step methods that use optimization approaches to learn combination parameters have high computational complexity, due to the fact that they are generally modeled as a semidefinite programming (SDP) problem, a quadratically constrained quadratic programming (QCQP) problem, or a second-order cone programming (SOCP) problem. These problems are much harder to solve than a quadratic programming (QP) problem used in the case of the canonical SVM.

Two-step methods update the combination function parameters and the base learner parameters in an alternating manner. The combination function parameters are generally updated by solving an optimization problem or using a closed-form update rule. Updating the base learner parameters usually requires training a kernel-based learner using the combined kernel. For example, they can be modeled as a semi-infinite linear programming (SILP) problem, which uses a generic linear programming (LP) solver and a canonical SVM solver in the inner loop.

3. Multiple Kernel Learning Algorithms

In this section, we categorize the existing MKL algorithms in the literature into 12 groups depending on the six key properties discussed in Section 2. We first give a summarizing table (see Tables 1 and 2) containing 49 representative references and then give a more detailed discussion of each group in a separate section reviewing a total of 96 references.

3.1 Fixed Rules

Fixed rules obtain $k_\eta(\cdot,\cdot)$ using $f_\eta(\cdot)$ and then train a canonical kernel machine with the kernel matrix calculated using $k_\eta(\cdot,\cdot)$. For example, we can obtain a valid kernel by taking the *summation* or *multiplication* of two valid kernels (Cristianini and Shawe-Taylor, 2000):

$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = k_1(\mathbf{x}_i^1,\mathbf{x}_j^1) + k_2(\mathbf{x}_i^2,\mathbf{x}_j^2)$$
$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = k_1(\mathbf{x}_i^1,\mathbf{x}_j^1)k_2(\mathbf{x}_i^2,\mathbf{x}_j^2). \qquad (1)$$

We know that a matrix $\mathbf{K}$ is positive semidefinite if and only if $\upsilon^\top\mathbf{K}\upsilon \geq 0$, for all $\upsilon \in \mathbb{R}^N$. Trivially, we can see that $k_1(\mathbf{x}_i^1,\mathbf{x}_j^1) + k_2(\mathbf{x}_i^2,\mathbf{x}_j^2)$ gives a positive semidefinite kernel matrix:

$$\upsilon^\top\mathbf{K}_\eta\upsilon = \upsilon^\top(\mathbf{K}_1+\mathbf{K}_2)\upsilon = \upsilon^\top\mathbf{K}_1\upsilon + \upsilon^\top\mathbf{K}_2\upsilon \geq 0$$

and $k_1(\mathbf{x}_i^1,\mathbf{x}_j^1)k_2(\mathbf{x}_i^2,\mathbf{x}_j^2)$ also gives a positive semidefinite kernel due to the fact that the element-wise product between two positive semidefinite matrices results in another positive semidefinite matrix:

$$\upsilon^\top\mathbf{K}_\eta\upsilon = \upsilon^\top(\mathbf{K}_1\odot\mathbf{K}_2)\upsilon \geq 0.$$

We can apply the rules in (1) recursively to obtain the rules for more than two kernels. For example, the summation or multiplication of P kernels is also a valid kernel:

$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = \sum_{m=1}^{P} k_m(\mathbf{x}_i^m,\mathbf{x}_j^m)$$
$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = \prod_{m=1}^{P} k_m(\mathbf{x}_i^m,\mathbf{x}_j^m).$$

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.

We need to calculate the similarity between pairs of objects such as genes or proteins especially in bioinformatics applications. *Pairwise kernels* are proposed to express the similarity between pairs in terms of similarities between individual objects. Two pairs are said to be similar when each object in one pair is similar to one object in the other pair. This approach can be encoded as a pairwise kernel using a kernel function between individual objects, called the *genomic kernel* (Ben-Hur and Noble, 2005), as follows:

$$k^P(\{\mathbf{x}_i^a,\mathbf{x}_j^a\},\{\mathbf{x}_i^b,\mathbf{x}_j^b\}) = k(\mathbf{x}_i^a,\mathbf{x}_i^b)k(\mathbf{x}_j^a,\mathbf{x}_j^b) + k(\mathbf{x}_i^a,\mathbf{x}_j^b)k(\mathbf{x}_j^a,\mathbf{x}_i^b).$$

Ben-Hur and Noble (2005) combine pairwise kernels in two different ways: (a) using an unweighted sum of different pairwise kernels:

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

and (b) using an unweighted sum of different genomic kernels in the pairwise kernel:

$$\begin{aligned}
&k_\eta^P(\{\mathbf{x}_i^a,\mathbf{x}_j^a\},\{\mathbf{x}_i^b,\mathbf{x}_j^b\})\\
&\quad= \left(\sum_{m=1}^{P} k_m(\mathbf{x}_i^a,\mathbf{x}_i^b)\right)\left(\sum_{m=1}^{P} k_m(\mathbf{x}_j^a,\mathbf{x}_j^b)\right) + \left(\sum_{m=1}^{P} k_m(\mathbf{x}_i^a,\mathbf{x}_j^b)\right)\left(\sum_{m=1}^{P} k_m(\mathbf{x}_j^a,\mathbf{x}_i^b)\right)\\
&\quad= k_\eta(\mathbf{x}_i^a,\mathbf{x}_i^b)k_\eta(\mathbf{x}_j^a,\mathbf{x}_j^b) + k_\eta(\mathbf{x}_i^a,\mathbf{x}_j^b)k_\eta(\mathbf{x}_j^a,\mathbf{x}_i^b).
\end{aligned}$$

The combined pairwise kernels improve the classification performance for protein-protein interaction prediction task.

Sec.	Representative References	Learning Method	Functional Form	Target Function	Training Method	Base Learner	Computational Complexity
3.1	Pavlidis et al. (2001)	Fixed	Lin. (unwei.)	None	1-step (seq.)	SVM	QP
	Ben-Hur and Noble (2005)	Fixed	Lin. (unwei.)	None	1-step (seq.)	SVM	QP
3.2	de Diego et al. (2004, 2010a)	Heuristic	Nonlinear	Val. error	2-step	SVM	QP
	Moguerza et al. (2004); de Diego et al. (2010a)	Heuristic	Data-dep.	None	1-step (seq.)	SVM	QP
	Tanabe et al. (2008)	Heuristic	Lin. (convex)	None	1-step (seq.)	SVM	QP
	Qiu and Lane (2009)	Heuristic	Lin. (convex)	None	1-step (seq.)	SVR	QP
	Qiu and Lane (2009)	Heuristic	Lin. (convex)	None	1-step (seq.)	SVM	QP
3.3	Lanckriet et al. (2004a)	Optim.	Lin. (linear)	Similarity	1-step (seq.)	SVM	SDP+QP
	Igel et al. (2007)	Optim.	Lin. (linear)	Similarity	1-step (seq.)	SVM	Grad.+QP
	Cortes et al. (2010a)	Optim.	Lin. (linear)	Similarity	1-step (seq.)	SVM	Mat. Inv.+QP
3.4	Lanckriet et al. (2004a)	Optim.	Lin. (conic)	Similarity	1-step (seq.)	SVM	QCQP+QP
	Kandola et al. (2002)	Optim.	Lin. (conic)	Similarity	1-step (seq.)	SVM	QP+QP
	Cortes et al. (2010a)	Optim.	Lin. (conic)	Similarity	1-step (seq.)	SVM	QP+QP
3.5	He et al. (2008)	Optim.	Lin. (convex)	Similarity	1-step (seq.)	SVM	QP+QP
	Tanabe et al. (2008)	Optim.	Lin. (convex)	Similarity	1-step (seq.)	SVM	QP+QP
	Ying et al. (2009)	Optim.	Lin. (convex)	Similarity	1-step (seq.)	SVM	Grad.+QP
3.6	Lanckriet et al. (2002)	Optim.	Lin. (linear)	Str. risk	1-step (seq.)	SVM	SDP+QP
	Qiu and Lane (2005)	Optim.	Lin. (linear)	Str. risk	1-step (seq.)	SVR	SDP+QP
	Conforti and Guido (2010)	Optim.	Lin. (linear)	Str. risk	1-step (seq.)	SVM	SDP+QP
3.7	Lanckriet et al. (2004a)	Optim.	Lin. (conic)	Str. risk	1-step (seq.)	SVM	QCQP+QP
	Fung et al. (2004)	Optim.	Lin. (conic)	Str. risk	2-step	KFDA	QP+Mat. Inv.
	Tsuda et al. (2004)	Optim.	Lin. (conic)	Str. risk	2-step	KFDA	Grad.+Mat. Inv.
	Qiu and Lane (2005)	Optim.	Lin. (conic)	Str. risk	1-step (seq.)	SVR	QCQP+QP
	Varma and Ray (2007)	Optim.	Lin. (conic)	Str. risk	1-step (sim.)	SVM	SOCP
	Varma and Ray (2007)	Optim.	Lin. (conic)	Str. risk	2-step	SVM	Grad.+QP
	Cortes et al. (2009)	Optim.	Lin. (conic)	Str. risk	2-step	KRR	Grad.+Mat. Inv.
	Kloft et al. (2010a)	Optim.	Lin. (conic)	Str. risk	2-step	SVM	Newton+QP
	Xu et al. (2010b)	Optim.	Lin. (conic)	Str. risk	1-step (sim.)	SVM	Grad.
	Kloft et al. (2010b); Xu et al. (2010a)	Optim.	Lin. (conic)	Str. risk	2-step	SVM	Analytical+QP
	Conforti and Guido (2010)	Optim.	Lin. (conic)	Str. risk	1-step (seq.)	SVM	QCQP+QP

Table 1: Representative MKL algorithms.

Sec.	Representative References	Learning Method	Functional Form	Target Function	Training Method	Base Learner	Computational Complexity
3.8	Bousquet and Herrmann (2003)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	Grad.+QP
	Bach et al. (2004)	Optim.	Lin. (convex)	Str. risk	1-step (sim.)	SVM	SOCP
	Sonnenburg et al. (2006a,b)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	LP+QP
	Kim et al. (2006)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	KFDA	SDP+Mat. Inv.
	Ye et al. (2007a)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	RKDA	SDP+Mat. Inv.
	Ye et al. (2007b)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	RKDA	QCQP+Mat. Inv.
	Ye et al. (2008)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	RKDA	SILP+Mat. Inv.
	Rakotomamonjy et al. (2007, 2008)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	Grad.+QP
	Chapelle and Rakotomamonjy (2008)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	QP+QP
	Kloft et al. (2010b); Xu et al. (2010a)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	Analytical+QP
	Conforti and Guido (2010)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	SVM	QCQP+QP
3.9	Lee et al. (2007)	Optim.	Nonlinear	Str. risk	1-step (sim.)	SVM	QP
	Varma and Babu (2009)	Optim.	Nonlinear	Str. risk	2-step	SVM	Grad.+QP
	Cortes et al. (2010b)	Optim.	Nonlinear	Str. risk	2-step	KRR	Grad.+Mat. Inv.
3.10	Lewis et al. (2006b)	Optim.	Data-dep.	Str. risk	1-step (sim.)	SVM	QP
	Gönen and Alpaydın (2008)	Optim.	Data-dep.	Str. risk	2-step	SVM	Grad.+QP
	Yang et al. (2009a)	Optim.	Data-dep.	Str. risk	2-step	SVM	Grad.+QP
	Yang et al. (2009b, 2010)	Optim.	Data-dep.	Str. risk	2-step	SVM	SILP+QP
3.11	Girolami and Rogers (2005)	Bayesian	Lin. (conic)	Likelihood	Inference	KRR	Approximation
	Girolami and Zhong (2007)	Bayesian	Lin. (conic)	Likelihood	Inference	GP	Approximation
	Christoudias et al. (2009)	Bayesian	Data-dep.	Likelihood	Inference	GP	Approximation
3.12	Bennett et al. (2002)	Boosting	Data-dep.	Str. risk	$P \times$ 1-step	KRR	Mat. Inv.
	Crammer et al. (2003)	Boosting	Lin. (conic)	Str. risk	$P \times$ 1-step	Percept.	Eigenvalue Prob.
	Bi et al. (2004)	Boosting	Lin. (linear)	Str. risk	$P \times$ 1-step	SVM	QP

Table 2: Representative MKL algorithms (continued).

3.2 Heuristic Approaches

de Diego et al. (2004, 2010a) 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:

$$\begin{aligned}
k_\eta(\mathbf{x}_i, \mathbf{x}_j) &= \frac{1}{2}(k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) + k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)) + \tau y_i y_j |k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) - k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)| \\
k_\eta(\mathbf{x}_i, \mathbf{x}_j) &= \frac{1}{2}(k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) + k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)) + \tau y_i y_j (k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) - k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)) \\
\mathbf{K}_\eta &= \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + \tau(\mathbf{K}_1 - \mathbf{K}_2)(\mathbf{K}_1 - \mathbf{K}_2)
\end{aligned}$$

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) and the first two functions do not ensure having positive semidefinite kernel matrices. It is also possible to combine more than two kernel functions by applying these rules recursively.

Moguerza et al. (2004) and de Diego et al. (2010a) propose a matrix functional form of combining kernels:

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

where $\eta_m(\cdot, \cdot)$ assigns a weight to $k_m(\cdot, \cdot)$ according to $\mathbf{x}_i$ and $\mathbf{x}_j$. They propose different heuristics to estimate the weighing function values using conditional class probabilities, $\Pr(y_i = y_j | \mathbf{x}_i)$ and $\Pr(y_j = y_i | \mathbf{x}_j)$, calculated with a nearest-neighbor approach. However, each kernel function corresponds to a different neighborhood and $\eta_m(\cdot, \cdot)$ is calculated on the neighborhood induced by $k_m(\cdot, \cdot)$. For an unlabeled data instance $\mathbf{x}$, they take its class label once as $+1$ and once as -1, calculate the discriminant values $f(\mathbf{x}|y = +1)$ and $f(\mathbf{x}|y = -1)$, and assign it to the class that has more confidence in its decision (i.e., by selecting the class label with greater $yf(\mathbf{x}|y)$ value). de Diego et al. (2010b) use this method to fuse information from several feature representations for face verification. Combining kernels in a data-dependent manner outperforms the classical fusion techniques such as feature-level and score-level methods in their experiments.

We can also use a linear combination instead of a data-dependent combination and formulate the combined kernel function as follows:

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

where we select the kernel weights by looking at the performance values obtained by each kernel separately. For example, Tanabe et al. (2008) propose the following rule in order to choose the kernel weights for classification problems:

$$\eta_m = \frac{\pi_m - \delta}{\sum\limits_{h=1}^{P} (\pi_h - \delta)}$$

where π_m is the accuracy obtained using only $\mathbf{K}_m$, and δ 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 problems:

$$\eta_m = \frac{R_m}{\sum\limits_{h=1}^{P} R_h} \quad \forall m$$

$$\eta_m = \frac{\sum\limits_{h=1}^{P} M_h - M_m}{(P-1)\sum\limits_{h=1}^{P} M_h} \quad \forall m$$

where R_m is the Pearson correlation coefficient between the true outputs and the predicted labels generated by the regressor using the kernel matrix $\mathbf{K}_m$, and M_m is the mean square error generated by the regressor using the kernel matrix $\mathbf{K}_m$. These three heuristics find a convex combination of the input kernels as the combined kernel.

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:

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

where $\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F = \sum_{i=1}^{N} \sum_{j=1}^{N} k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)$. This similarity measure can be seen as the cosine of the angle between $\mathbf{K}_1$ and $\mathbf{K}_2$. $\mathbf{y}\mathbf{y}^\top$ can be defined as *ideal kernel* for a binary classification task, and the alignment between a kernel and the ideal kernel becomes

$$\text{A}(\mathbf{K}, \mathbf{y}\mathbf{y}^\top) = \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^\top \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{y}\mathbf{y}^\top, \mathbf{y}\mathbf{y}^\top \rangle_F}} = \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^\top \rangle_F}{N\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F}}.$$

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

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

$$\eta_m = \frac{\text{A}(\mathbf{K}_m, \mathbf{y}\mathbf{y}^\top)}{\sum\limits_{h=1}^{P} \text{A}(\mathbf{K}_h, \mathbf{y}\mathbf{y}^\top)} \quad \forall m \tag{2}$$

where we obtain the combined kernel as a convex combination of the input kernels.

3.3 Similarity Optimizing Linear Approaches with Arbitrary Kernel Weights

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

$$\begin{aligned}
\text{maximize} \quad & \operatorname{A}(\mathbf{K}_{\eta}^{\text{tra}}, \mathbf{y}\mathbf{y}^{\top}) \\
\text{with respect to} \quad & \mathbf{K}_{\eta} \in \mathbb{S}^N \\
\text{subject to} \quad & \operatorname{tr}\left(\mathbf{K}_{\eta}\right) = 1 \\
& \mathbf{K}_{\eta} \succeq 0
\end{aligned}$$

where the trace of the combined kernel matrix is arbitrarily set to 1. This problem can be converted into the following SDP problem using arbitrary kernel weights in the combination:

$$\begin{aligned}
\text{maximize} \quad & \left\langle \sum_{m=1}^{P} \eta_m \mathbf{K}_m^{\text{tra}}, \mathbf{y}\mathbf{y}^{\top} \right\rangle_F \\
\text{with respect to} \quad & \eta \in \mathbb{R}^P, \;\; \mathbf{A} \in \mathbb{S}^N \\
\text{subject to} \quad & \operatorname{tr}(\mathbf{A}) \leq 1 \\
& \begin{pmatrix} \mathbf{A} & \sum_{m=1}^{P} \eta_m \mathbf{K}_m^{\top} \\ \sum_{m=1}^{P} \eta_m \mathbf{K}_m & \mathbf{I} \end{pmatrix} \succeq 0 \\
& \sum_{m=1}^{P} \eta_m \mathbf{K}_m \succeq 0.
\end{aligned}$$

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

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

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.

Cortes et al. (2010a) give a different kernel alignment definition, which they call *centered-kernel alignment*. The empirical centered-alignment of two kernels is calculated as follows:

$$\operatorname{CA}(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1^c, \mathbf{K}_2^c \rangle_F}{\sqrt{\langle \mathbf{K}_1^c, \mathbf{K}_1^c \rangle_F \langle \mathbf{K}_2^c, \mathbf{K}_2^c \rangle_F}}$$

where $\mathbf{K}^c$ is the centered version of $\mathbf{K}$ and can be calculated as

$$\mathbf{K}^c = \mathbf{K} - \frac{1}{N}\mathbf{1}\mathbf{1}^{\top}\mathbf{K} - \frac{1}{N}\mathbf{K}\mathbf{1}\mathbf{1}^{\top} + \frac{1}{N^2}(\mathbf{1}^{\top}\mathbf{K}\mathbf{1})\mathbf{1}\mathbf{1}^{\top}$$

where **1** is the vector of ones with proper dimension. Cortes et al. (2010a) also propose to optimize the centered-kernel alignment as follows:

$$\begin{aligned} \text{maximize } & \operatorname{CA}(\mathbf{K}_\eta, \mathbf{y}\mathbf{y}^\top) \\ \text{with respect to } & \eta \in \mathcal{M} \end{aligned} \tag{3}$$

where $\mathcal{M} = \{\eta \colon \|\eta\|_2 = 1\}$. This optimization problem (3) has an analytical solution:

$$\eta = \frac{\mathbf{M}^{-1}\mathbf{a}}{\|\mathbf{M}^{-1}\mathbf{a}\|_2} \tag{4}$$

where $\mathbf{M} = \{\langle \mathbf{K}_m^c, \mathbf{K}_h^c \rangle_F\}_{m,h=1}^P$ and $\mathbf{a} = \{\langle \mathbf{K}_m^c, \mathbf{y}\mathbf{y}^\top \rangle_F\}_{m=1}^P$.

3.4 Similarity Optimizing Linear Approaches with Nonnegative Kernel Weights

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:

$$\operatorname{A}(\mathbf{K}_\eta, \mathbf{y}\mathbf{y}^\top) = \frac{\sum_{m=1}^P \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^\top \rangle_F}{N \sqrt{\sum_{m=1}^P \sum_{h=1}^P \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F}}.$$

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

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

and this problem is equivalent to

$$\begin{aligned} \text{maximize } & \sum_{m=1}^P \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^\top \rangle_F \\ \text{with respect to } & \eta \in \mathbb{R}_+^P \\ \text{subject to } & \sum_{m=1}^P \sum_{h=1}^P \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F = c. \end{aligned}$$

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

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

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:

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^\top \rangle_F - \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - \lambda \sum_{m=1}^{P} \eta_m^2 \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \end{aligned}$$

where we only learn the kernel combination weights.

Lanckriet et al. (2004a) restrict the kernel weights to be nonnegative and their SDP formulation reduces to the following QCQP problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m^{\text{tra}}, \mathbf{y}\mathbf{y}^\top \rangle_F \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F \leq 1. \end{aligned} \tag{5}$$

Cortes et al. (2010a) also restrict the kernel weights to be nonnegative by changing the definition of $\mathcal{M}$ in (3) to $\{\eta \colon \|\eta\|_2 = 1, \ \eta \in \mathbb{R}_+^P\}$ and obtain the following QP:

$$\begin{aligned} \text{minimize} \quad & \mathbf{v}^\top \mathbf{M} \mathbf{v} - 2\mathbf{v}^\top \mathbf{a} \\ \text{with respect to} \quad & \mathbf{v} \in \mathbb{R}_+^P \end{aligned} \tag{6}$$

where the kernel weights are given by $\eta = \mathbf{v}/\|\mathbf{v}\|_2$.

3.5 Similarity Optimizing Linear Approaches with Kernel Weights on a Simplex

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, using the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \langle \mathbf{K}_\eta - \mathbf{y}\mathbf{y}^\top, \mathbf{K}_\eta - \mathbf{y}\mathbf{y}^\top \rangle_F^2 \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned}$$

This problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - 2 \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^\top \rangle_F \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned} \tag{7}$$

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

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

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

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

This method gives 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 KL divergence between the combined kernel matrix and the optimal kernel matrix:

$$\begin{aligned} \text{minimize} \quad & \text{KL}(\mathcal{N}(\mathbf{0}, \mathbf{K}_\eta) \| \mathcal{N}(\mathbf{0}, \mathbf{y}\mathbf{y}^\top)) \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1 \end{aligned}$$

where $\mathbf{0}$ is the vector of zeros with proper dimension. The kernel combinations weights can be optimized using a projected gradient-descent method.

3.6 Structural Risk Optimizing Linear Approaches with Arbitrary Kernel Weights

Lanckriet et al. (2002) follow a direct approach in order to optimize the unrestricted kernel combination weights. The *implausibility* of a kernel matrix, $\omega(\mathbf{K})$, 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 the ℓ_1-norm on slack variables):

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

The combined kernel matrix is selected from the following set:

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

where the selected kernel matrix 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

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

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

$$\begin{aligned} \text{minimize} \quad & t \\ \text{with respect to} \quad & \eta \in \mathbb{R}^P, \ t \in \mathbb{R}, \ \lambda \in \mathbb{R}, \ \nu \in \mathbb{R}_+^N, \ \delta \in \mathbb{R}_+^N \\ \text{subject to} \quad & \operatorname{tr}\left(\mathbf{K}_\eta\right) = c \\ & \begin{pmatrix} (\mathbf{y}\mathbf{y}^\top) \odot \mathbf{K}_\eta^{\text{tra}} & \mathbf{1} + \nu - \delta + \lambda\mathbf{y} \\ (\mathbf{1} + \nu - \delta + \lambda\mathbf{y})^\top & t - 2C\delta^\top\mathbf{1} \end{pmatrix} \succeq 0 \\ & \mathbf{K}_\eta \succeq 0. \end{aligned}$$

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

Lanckriet et al. (2004a,c) consider 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 an SDP formulation. This gives better results than SVMs trained with each kernel in nine out of 13 experiments. Qiu and Lane (2005) extends ε-tube SVR to a QCQP formulation for regression problems. Conforti and Guido (2010) propose another SDP formulation that removes trace restriction on the combined kernel matrix and introduces constraints over the kernel weights for an inductive setting.

3.7 Structural Risk Optimizing Linear Approaches with Nonnegative Kernel Weights

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

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

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

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

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 also a QCQP formulation of regression using ε-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. (2004a) combine two different kernels obtained from heterogeneous information sources, namely, bag-of-words and graphical representations, on the Reuters-21578 data set. Combining these two kernels with positive weights outperforms the single-kernel results obtained with SVM on four tasks out of five. Lanckriet et al. (2004b) use a QCQP formulation to integrate multiple kernel functions calculated on heterogeneous views of the 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. The 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.

A similar result about unweighted and weighted linear kernel combinations is also obtained by Lewis et al. (2006a). They 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.

Fung et al. (2004) propose an iterative algorithm using the kernel Fisher discriminant analysis as the base learner to combine heterogeneous kernels 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. They use the sigmoid function for error 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. They 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 that 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 using the minimal upper bound of the Vapnik-Chervonenkis dimension as the target function.

Varma and Ray (2007) propose a formulation for combining kernels using a linear combination with regularized nonnegative weights. The regularization on the kernel combination weights is achieved by adding a term to the objective function and integrating a set of constraints. The primal optimization problem with these two modifications can be given as

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

where $\Phi_\eta(\cdot)$ corresponds to the feature space that implicitly constructs the combined kernel function $k_\eta(\mathbf{x}_i,\mathbf{x}_j) = \sum_{m=1}^{P}\eta_m k_m(\mathbf{x}_i^m,\mathbf{x}_j^m)$ and $\mathbf{w}_\eta$ is the vector of weight coefficients assigned to $\Phi_\eta(\cdot)$. The parameters $\mathbf{A} \in \mathbb{R}^{R\times P}$, $\mathbf{p} \in \mathbb{R}^R$, and $\sigma \in \mathbb{R}^P$ encode our prior information about the kernel weights. For example, assigning higher σ_i values to some of the kernels effectively eliminates them by assigning zero weights to them. The corresponding dual formulation is derived as the following SOCP problem:

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

Instead of solving this SOCP problem directly, Varma and Ray (2007) also propose an alternating optimization problem that performs projected gradient updates for kernel weights and solves a QP problem to find the support vector coefficients at each iteration. The primal optimization problem for given η is written as

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

and the corresponding dual optimization problem is

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

The gradients with respect to the kernel weights are calculated as

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

and these gradients are used to update the kernel weights while considering nonnegativity and other constraints.

Usually, the kernel weights are constrained by a trace or the ℓ_1-norm regularization. Cortes et al. (2009) discuss the suitability of the ℓ_2-norm for MKL. They combine kernels with ridge regression using the ℓ_2-norm regularization over the kernel weights. They conclude that using the ℓ_1-norm improves the performance for a small number of kernels, but degrades the performance when combining a large number of kernels. However, the ℓ_2-norm never decreases the performance and increases it significantly for larger sets of candidate kernels. Yan et al. (2009) compare the ℓ_1-norm and the ℓ_2-norm for image and video classification tasks, and conclude that the ℓ_2-norm should be used when the combined kernels carry complementary information.

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

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

and they solve this problem using alternative optimization strategies based on Newton-descent and cutting planes. Xu et al. (2010b) add an entropy regularization term instead of constraining the norm of the kernel weights and derive an efficient and smooth optimization framework based on Nesterov's method.

Kloft et al. (2010b) and Xu et al. (2010a) propose an efficient optimization method for arbitrary ℓ_p-norms with $p \geq 1$. Although they approach the problem from different perspectives, they find the same closed-form solution for updating the kernel weights at each iteration. Kloft et al. (2010b) use a block coordinate-descent method and Xu et al. (2010a) use the equivalence between group Lasso and MKL, as shown by Bach (2008) to derive the update equation. Both studies formulate an alternating optimization method that solves an SVM at each iteration and update the kernel weights as follows:

$$\eta_m = \frac{\|\mathbf{w}_m\|_2^{\frac{2}{p+1}}}{\left(\sum\limits_{h=1}^{P} \|\mathbf{w}_h\|_2^{\frac{2p}{p+1}} \right)^{\frac{1}{p}}} \tag{8}$$

where $\|\mathbf{w}_m\|_2^2 = \eta_m^2 \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$ from the duality conditions.

When we restrict the kernel weights to be nonnegative, the SDP formulation of Conforti and Guido (2010) reduces to a QCQP problem.

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.

Another possibility is to allow only binary η_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

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

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

3.8 Structural Risk Optimizing Linear Approaches with Kernel Weights on a Simplex

We can think of kernel combination as a weighted average of kernels and consider $\eta \in \mathbb{R}_+^P$ 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 because in combination, we want kernels to be useful by themselves and complementary. In a web page classification experiment, they show that 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, θ:

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

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

where α is obtained by solving the canonical SVM optimization problem and β is obtained by solving the QP problem defined by Vapnik (1998). These derivatives can be used to optimize the individual parameters (e.g., scaling coefficient) on each feature 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 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

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

where $\mathbf{w}_{\eta}$ is the weight vector obtained using $\mathbf{K}_{\eta}$ in training. In an iterative manner, an SVM is trained to obtain α, then η is updated using the calculated gradient while considering nonnegativity (i.e., $\eta \in \mathbb{R}_+^P$) 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.

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

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

where the feature space constructed using $\Phi_m(\cdot)$ has the dimensionality S_m and the weight d_m. When we consider this optimization problem as an SOCP problem, we obtain the following dual formulation:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\gamma^2 - \sum_{i=1}^{N}\alpha_i \\
\text{with respect to} \quad & \gamma \in \mathbb{R},\ \alpha \in \mathbb{R}_+^N \\
\text{subject to} \quad & \gamma^2 d_m^2 \geq \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \quad \forall m \\
& \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i
\end{aligned} \tag{9}$$

where we again get the optimal kernel weights from the optimal dual variables and the weights satisfy $\sum_{m=1}^{P} d_m^2\eta_m = 1$. The dual problem is exactly equivalent to the QCQP formulation of Lanckriet et al. (2004a) when we take $d_m = \sqrt{\operatorname{tr}(\mathbf{K}_m)/c}$. The advantage of the SOCP formulation is that Bach et al. (2004) devise an SMO-like algorithm by adding a Moreau-Yosida regularization term, $1/2\sum_{m=1}^{P} a_m^2\|\mathbf{w}_m\|_2^2$, to the primal objective function and deriving the corresponding dual formulation. Using the 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 the area under the ROC curve. Özen et al. (2009) use the formulation of Bach et al. (2004) 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):

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

and convert this problem into the following SILP problem:

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

where the problem has infinitely many constraints due to the possible values of α.

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

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 an SDP problem inspired by Kim et al. (2006) for learning an optimal kernel over a convex set of candidate kernels for RKDA. 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 QCQP and SILP formulations equivalent to the previous SDP problem in order to reduce the time complexity. These three formulations are directly applicable to multiclass classification because it uses RKDA as the base learner.

De Bie et al. (2007) derive a QCQP formulation of one-class classification using a convex combination of multiple kernels. In order to prevent the combined kernel from overfitting, they also propose a modified mathematical model that defines 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 solution.

Zien and Ong (2007) develop a QCQP formulation and convert this formulation in two different SILP problems for multiclass classification. 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 MKL 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

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

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

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

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

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

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

$$\begin{aligned} \text{minimize} \;\; & J(\eta) \\ \text{with respect to} \;\; & \eta \in \mathbb{R}_+^P \\ \text{subject to} \;\; & \sum_{m=1}^{P} \eta_m = 1. \end{aligned} \tag{10}$$

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

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

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 an SILP formulation. Both algorithms use a canonical SVM solver in order to calculate α at each step. The difference is that they use different updating procedures for η, 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, one-class and multiclass classification (Rakotomamonjy et al., 2008).

Chapelle and Rakotomamonjy (2008) propose a second order method, called HESSIANMKL, extending SIMPLEMKL. HESSIANMKL updates kernel weights at each iteration using a constrained Newton step found by solving a QP problem. Chapelle and Rakotomamonjy (2008) show that HESSIANMKL converges faster than SIMPLEMKL.

Xu et al. (2009a) propose a hybrid method that combines the SILP formulation of Sonnenburg et al. (2006b) and SIMPLEMKL of Rakotomamonjy et al. (2008). The 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.

The alternating optimization method proposed by Kloft et al. (2010b) and Xu et al. (2010a) learns a convex combination of kernels when we use the ℓ_1-norm for regularizing the kernel weights. When we take $p = 1$, the update equation in (8) becomes

$$\eta_m = \frac{\|\mathbf{w}_m\|_2}{\sum\limits_{h=1}^{P} \|\mathbf{w}_h\|_2}. \tag{11}$$

The SDP formulation of Conforti and Guido (2010) reduces to a QCQP problem when we use a convex combination of the base kernels.

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

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

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

Micchelli and Pontil (2005) try to learn the 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 selecting kernels from basically an infinite set, instead of a finite number of basic kernels.

Instead of selecting kernels from a predefined finite set, we can increase the number of candidate kernels in an iterative manner. We can basically select kernels from an uncountably infinite set constructed by considering base kernels with different kernel parameters (Özöğür-Akyüz and Weber, 2008; Gehler and Nowozin, 2008). Gehler and Nowozin (2008) propose a forward selection algorithm that finds the kernel weights for a fixed size of candidate kernels 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, 2010) follow this idea and derive an MKL method by changing the mathematical model used by Rakotomamonjy et al. (2007). Saketha Nath et al. (2010) propose another MKL method that considers the group structure between the kernels and this method assumes that every kernel group carries important information. The proposed formulation enforces the ℓ_∞-norm at the group level and the ℓ_1-norm within each group. By doing this, each group is used in the final learner, but sparsity is promoted among kernels in each group. They formulate the problem as an SCOP problem and give a highly efficient optimization algorithm that uses a mirror-descent approach.

Subrahmanya and Shin (2010) 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.

Most of the structural risk optimizing linear approaches can be casted into a general framework (Kloft et al., 2010a,b). The unified optimization problem with the Tikhonov regularization can be written as

$$\begin{aligned} &\text{minimize } \frac{1}{2} \sum_{m=1}^{P} \frac{\|\mathbf{w}_m\|_2^2}{\eta_m} + C \sum_{i=1}^{N} L\left(\sum_{m=1}^{P} \langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m) \rangle + b, y_i \right) + \mu \|\eta\|_p^p \\ &\text{with respect to } \mathbf{w}_m \in \mathbb{R}^{S_m},\ b \in \mathbb{R},\ \eta \in \mathbb{R}_+^P \end{aligned}$$

where $L(\cdot,\cdot)$ is the loss function used. Alternatively, we can use the Ivanov regularization instead of the Tikhonov regularization by integrating an additional constraint into the optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{P}\frac{\|\mathbf{w}_m\|_2^2}{\eta_m} + C\sum_{i=1}^{N} L\left(\sum_{m=1}^{P}\langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b, y_i\right) \\ \text{with respect to} \quad & \mathbf{w}_m \in \mathbb{R}^{S_m},\ b \in \mathbb{R},\ \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \|\eta\|_p^p \leq 1. \end{aligned}$$

Figure 1 lists the MKL algorithms that can be casted into the general framework described above. Zien and Ong (2007) show that their formulation is equivalent to those of Bach et al. (2004) and Sonnenburg et al. (2006a,b). Using unified optimization problems given above and the results of Zien and Ong (2007), Kloft et al. (2010a,b) show that the formulations with $p = 1$ in Figure 1 fall into the same equivalence class and introduce a new formulation with $p \geq 1$. The formulation of Xu et al. (2010a) is also equivalent to those of Kloft et al. (2010a,b).

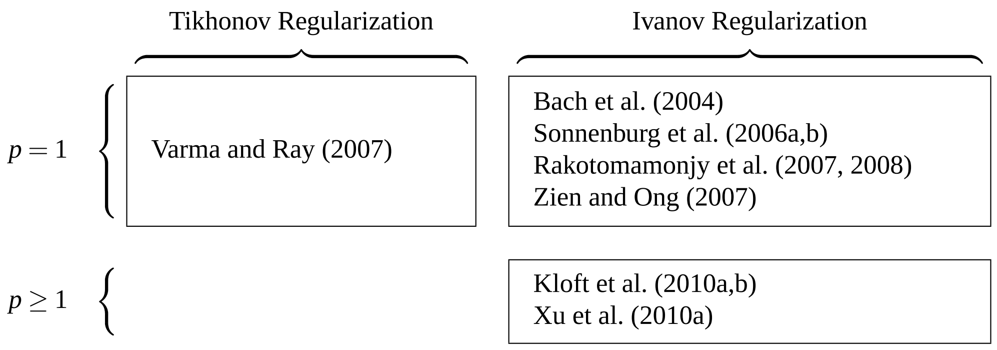

Figure 1: MKL algorithms that can be casted into the general framework described.

3.9 Structural Risk Optimizing Nonlinear Approaches

Ong et al. (2003) propose to learn 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. Hyperkernels are generalized to different machine learning problems such as binary classification, regression, and one-class classification (Ong and Smola, 2003; Ong et al., 2005). 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 SOCP problems in order to reduce the time complexity of the training phase.

Varma and Babu (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 regularizes both the hyperplane weights and the kernel combination weights. The loss function can be one of the classical loss functions, such as, hinge loss for classification, or ϵ-loss for regression. The proposed primal formulation applied to binary classification problem with hinge loss and the regularization function, $r(\cdot)$, can be written as

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

where $\Phi_\eta(\cdot)$ corresponds to the feature space that implicitly constructs the combined kernel function $k_\eta(\cdot,\cdot)$ and $\mathbf{w}_\eta$ is the vector of weight coefficients assigned to $\Phi_\eta(\cdot)$. 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 η is used as the target function:

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

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

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

The regularization function $r(\cdot)$ and $k_\eta(\cdot,\cdot)$ can be any differentiable function of η with continuous derivative. The gradient with respect to the kernel weights is calculated as

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

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

$$\begin{aligned} k_\eta^S(\mathbf{x}_i,\mathbf{x}_j) &= \sum_{m=1}^{D}\eta_m \exp\left(-\gamma_m\left(\mathbf{x}_i[m] - \mathbf{x}_j[m]\right)^2\right) \\ k_\eta^P(\mathbf{x}_i,\mathbf{x}_j) &= \prod_{m=1}^{D}\exp\left(-\eta_m\left(\mathbf{x}_i[m] - \mathbf{x}_j[m]\right)^2\right) = \exp\left(\sum_{m=1}^{D} -\eta_m\left(\mathbf{x}_i[m] - \mathbf{x}_j[m]\right)^2\right). \end{aligned}$$

They show that GMKL with $k_\eta^P(\cdot,\cdot)$ performs significantly better than SIMPLEMKL with $k_\eta^S(\cdot,\cdot)$. We see that using $k_\eta^P(\cdot,\cdot)$ 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. (2010b) develop a nonlinear kernel combination method based on KRR and polynomial combination of kernels. They propose to combine kernels as follows:

$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = \sum_{\mathbf{q}\in\mathcal{Q}} \eta_{q_1q_2\ldots q_P} k_1(\mathbf{x}_i^1,\mathbf{x}_j^1)^{q_1} k_2(\mathbf{x}_i^2,\mathbf{x}_j^2)^{q_2} \ldots k_P(\mathbf{x}_i^P,\mathbf{x}_j^P)^{q_P}$$

where $\mathcal{Q} = \{\mathbf{q}\colon \mathbf{q}\in\mathbb{Z}_+^P,\ \sum_{m=1}^P q_m \leq d\}$ and $\eta_{q_1q_2\ldots q_P} \geq 0$. The number of parameters to be learned is too large and the combined kernel is simplified in order to reduce the learning complexity:

$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = \sum_{\mathbf{q}\in\mathcal{R}} \eta_1^{q_1}\eta_2^{q_2}\ldots\eta_P^{q_P} k_1(\mathbf{x}_i^1,\mathbf{x}_j^1)^{q_1} k_2(\mathbf{x}_i^2,\mathbf{x}_j^2)^{q_2} \ldots k_P(\mathbf{x}_i^P,\mathbf{x}_j^P)^{q_P}$$

where $\mathcal{R} = \{\mathbf{q}\colon \mathbf{q}\in\mathbb{Z}_+^P,\ \sum_{m=1}^P q_m = d\}$ and $\eta\in\mathbb{R}^P$. For example, when $d=2$, the combined kernel function becomes

$$k_\eta(\mathbf{x}_i,\mathbf{x}_j) = \sum_{m=1}^P\sum_{h=1}^P \eta_m\eta_h k_m(\mathbf{x}_i^m,\mathbf{x}_j^m)k_h(\mathbf{x}_i^h,\mathbf{x}_j^h). \tag{12}$$

The combination weights are optimized using the following min-max optimization problem:

$$\underset{\eta\in\mathcal{M}}{\text{minimize}}\ \underset{\alpha\in\mathbb{R}^N}{\text{maximize}}\ -\alpha^\top(\mathbf{K}_\eta+\lambda\mathbf{I})\alpha + 2\mathbf{y}^\top\alpha$$

where $\mathcal{M}$ is a positive, bounded, and convex set. Two possible choices for the set $\mathcal{M}$ are the ℓ_1-norm and ℓ_2-norm bounded sets defined as

$$\mathcal{M}_1 = \{\eta\colon \eta\in\mathbb{R}_+^P,\ \|\eta-\eta_0\|_1 \leq \Lambda\} \tag{13}$$

$$\mathcal{M}_2 = \{\eta\colon \eta\in\mathbb{R}_+^P,\ \|\eta-\eta_0\|_2 \leq \Lambda\} \tag{14}$$

where η_0 and Λ are two model parameters. A projection-based gradient-descent algorithm can be used to solve this min-max optimization problem. At each iteration, α is obtained by solving a KRR problem with the current kernel matrix and η is updated with the gradients calculated using α while considering the bound constraints on η due to $\mathcal{M}_1$ or $\mathcal{M}_2$.

Lee et al. (2007) follow a different approach and combine kernels using a compositional method that constructs a $(P\times N)\times(P\times N)$ compositional kernel matrix. This matrix and the training instances replicated P times are used to train a canonical SVM.

3.10 Structural Risk Optimizing Data-Dependent Approaches

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) propose a data-dependent formulation called localized multiple kernel learning (LMKL) that combines kernels using weights calculated from a gating model. The proposed primal optimization problem is

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

where the gating model $\eta_m(\cdot|\cdot)$, parameterized by $\mathbf{V}$, assigns a weight to the feature space obtained with $\Phi_m(\cdot)$. 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:

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

where the combined kernel matrix is represented as

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

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

$$\eta_m(\mathbf{x}|\mathbf{V}) = \frac{\exp(\langle\mathbf{v}_m, \mathbf{x}^{\mathcal{G}}\rangle + v_{m0})}{\sum_{h=1}^{P}\exp(\langle\mathbf{v}_h, \mathbf{x}^{\mathcal{G}}\rangle + v_{h0})} \quad \forall m \tag{15}$$

where $\mathbf{V} = \{\mathbf{v}_m, v_{m0}\}_{m=1}^P$, $\mathbf{x}^{\mathcal{G}} \in \mathbb{R}^{D_{\mathcal{G}}}$ is the representation of the input instance in the feature space in which we learn the gating model and there are $P \times (D_{\mathcal{G}}+1)$ parameters where $D_{\mathcal{G}}$ is the dimensionality of the gating feature space. The softmax gating model uses kernels in a competitive manner and generally a single kernel is active for each input. We may also use the sigmoid function instead of softmax and thereby allow multiple kernels to be used in a cooperative manner:

$$\eta_m(\mathbf{x}|\mathbf{V}) = \frac{1}{\exp(-\langle\mathbf{v}_m, \mathbf{x}^{\mathcal{G}}\rangle - v_{m0})} \quad \forall m. \tag{16}$$

The gating model parameters are updated at each iteration by calculating $\partial J(\mathbf{V})/\partial\mathbf{V}$ and performing a gradient-descent step (Gönen and Alpaydın, 2008).

Inspired from LMKL, two methods that learn a data-dependent kernel function are used for image recognition applications (Yang et al., 2009a,b, 2010); they differ in their gating models that are constants rather than functions of the input. Yang et al. (2009a) divide the training set into clusters as a preprocessing step, and then cluster-specific kernel weights are learned using alternating optimization. The combined kernel function can be written as

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_{c_i}^m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \eta_{c_j}^m$$

where $\eta_{c_i}^m$ corresponds to the weight of kernel $k_m(\cdot, \cdot)$ in the cluster $\mathbf{x}_i$ belongs to. The kernel weights of the cluster that the test instance is assigned to are used in the testing phase. Yang et al. (2009b, 2010) use instance-specific kernel weights instead of cluster-specific weights. The corresponding combined kernel function is

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_i^m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \eta_j^m$$

where η_i^m corresponds to the weight of kernel $k_m(\cdot, \cdot)$ for $\mathbf{x}_i$ and these instance-specific weights are optimized using alternating optimization over the training set. In the testing phase, the kernel weights for a test instance are all taken to be equal.

3.11 Bayesian Approaches

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

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

where η is modeled with a Dirichlet prior and α 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 GPs. 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 GP coefficients and the kernel combination weights.

Similar to LMKL, Christoudias et al. (2009) develop a Bayesian approach for combining different feature representations in a data-dependent way under the GP framework. A common covariance function is obtained by combining the covariances of feature representations in a nonlinear manner. This formulation can identify the noisy data instances for each feature representation and prevent them from being used. Classification is performed using the standard GP approach with the common covariance function.

3.12 Boosting Approaches

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

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

The parameters $\{\alpha^m\}_{m=1}^P$ and b of the KRR model are learned using gradient-descent in the function space. The columns of the combined kernel matrix are generated on the fly from the heterogeneous kernels. Bi et al. (2004) develop column generation boosting methods for binary classification and regression problems. At 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:

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

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

4. Experiments

In order to compare several MKL algorithms, we perform 10 different experiments on four data sets that are composed of different feature representations. We use both the linear kernel and the Gaussian kernel in our experiments; we will give our results with the linear kernel first and then compare them with the results of the Gaussian kernel. The kernel matrices are normalized to unit diagonal before training.

4.1 Compared Algorithms

We implement two single-kernel SVM and 16 representative MKL algorithms in MATLAB[1] and solve the optimization problems with the MOSEK optimization software (Mosek, 2011).

We train SVMs on each feature representation singly and report the results of the one with the highest average validation accuracy, which will be referred as SVM (best). We also train an SVM on the concatenation of all feature representations, which will be referred as SVM (all).

RBMKL denotes rule-based MKL algorithms discussed in Section 3.1. RBMKL (mean) trains an SVM with the mean of the combined kernels. RBMKL (product) trains an SVM with the product of the combined kernels.

ABMKL denotes alignment-based MKL algorithms. For determining the kernel weights, ABMKL (ratio) uses the heuristic in (2) of Section 3.2 (Qiu and Lane, 2009), ABMKL (conic) solves the QCQP problem in (5) of Section 3.4 (Lanckriet et al., 2004a), and ABMKL (convex) solves the QP problem in (7) of Section 3.5 (He et al., 2008). In the second step, all methods train an SVM with the kernel calculated with these weights.

CABMKL denotes centered-alignment-based MKL algorithms. In the first step, CABMKL (linear) uses the analytical solution in (4) of Section 3.3 (Cortes et al., 2010a) and CABMKL (conic) solves

1. Implementations are available at `http://www.cmpe.boun.edu.tr/~gonen/mkl`.

the QP problem in (6) of Section 3.4 (Cortes et al., 2010a) for determining the kernel weights. In the second step, both methods train an SVM with the kernel calculated with these weights.

MKL is the original MKL algorithm of Bach et al. (2004) that is formulated as the SOCP problem in (9) of Section 3.8. SimpleMKL is the iterative algorithm of Rakotomamonjy et al. (2008) that uses projected gradient updates and trains SVMs at each iteration to solve the optimization problem in (10) of Section 3.8.

GMKL is the generalized MKL algorithm of Varma and Babu (2009) discussed in Section 3.9. In our implementation, $k_{\eta}(\cdot,\cdot)$ is the convex combination of base kernels and $r(\cdot)$ is taken as $1/2(\eta - \mathbf{1}/P)^{\top}(\eta - \mathbf{1}/P)$.

GLMKL denotes the group Lasso-based MKL algorithms proposed by Kloft et al. (2010b) and Xu et al. (2010a). GLMKL ($p = 1$) updates the kernel weights using (11) of Section 3.8 and learns a convex combination of the kernels. GLMKL ($p = 2$) updates the kernel weights setting $p = 2$ in (8) of Section 3.7 and learns a conic combination of the kernels.

NLMKL denotes the nonlinear MKL algorithm of Cortes et al. (2010b) discussed in Section 3.9 with the exception of replacing the KRR in the inner loop with an SVM as the base learner. NLMKL uses the quadratic kernel given in (12). NLMKL ($p = 1$) and NLMKL ($p = 2$) select the kernel weights from the sets $\mathcal{M}_1$ in (13) and $\mathcal{M}_2$ in (14), respectively. In our implementation, η_0 is taken as $\mathbf{0}$ and Λ is assigned to 1 arbitrarily.

LMKL denotes the localized MKL algorithm of Gönen and Alpaydın (2008) discussed in Section 3.10. LMKL (softmax) uses the softmax gating model in (15), whereas LMKL (sigmoid) uses the sigmoid gating model in (16). Both methods use the concatenation of all feature representations in the gating model.

4.2 Experimental Methodology

Our experimental methodology is as follows: Given a data set, if learning and test sets are not supplied separately, a random one-third is reserved as the test set and the remaining two-thirds is used as the learning set. If the learning set has more than 1000 data instances, it is resampled using 5×2 cross-validation to generate 10 training and validation sets, with stratification, otherwise, we use 30-fold cross-validation. The validation sets of all folds are used to optimize the common hyperparameter C (trying values 0.01, 0.1, 1, 10, and 100). 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. Their test accuracies, support vector percentages, active kernel[2] counts, and numbers of calls to the optimization toolbox for solving an SVM optimization problem or a more complex optimization problem[3] are measured; we report their averages and standard deviations. The active kernel count and the number of calls to the optimization toolbox for SVM (best) are taken as 1 and P, respectively, because it uses only one of the feature representations but needs to train the individual SVMs on all feature representations before choosing the best. Similarly, the active kernel count and the number of calls to the optimization toolbox for SVM (all) are taken as P and 1, respectively, because it uses all of the feature representations but trains a single SVM.

2. A kernel is *active*, if it needs to be calculated to make a prediction for an unseen test instance.
3. All algorithms except the MKL formulation of Bach et al. (2004), MKL, solve QP problems when they call the optimization toolbox, whereas MKL solves an SOCP problem.

The test accuracies and support vector percentages are compared using the 5×2 cv paired F test (Alpaydın, 1999) or the paired t test according to the resampling scheme used. The active kernel counts and the number of calls to the optimization toolbox are compared using the Wilcoxon's signed-rank test (Wilcoxon, 1945). For all statistical tests, the significance level, α, is taken as 0.05. We want to test if by combining kernels, we get accuracy higher than any of the single kernels. In the result tables, a superscript a denotes that the performance values of SVM (best) and the compared algorithm are statistically significantly different, where $\overline{\text{a}}$ and $\underline{\text{a}}$ denote that the compared algorithm has statistically significantly higher and lower average than SVM (best), respectively. Similarly, we want to test if an algorithm is better than a straightforward concatenation of the input features, SVM (all), and if it is better than fixed combination, namely, RBMKL (mean); for those, we use the superscripts b and c, respectively.

4.3 Protein Fold Prediction Experiments

We perform experiments on the Protein Fold (PROTEIN) prediction data set[4] from the MKL Repository, composed of 10 different feature representations and two kernels for 694 instances (311 for training and 383 for testing). The properties of these feature representations are summarized in Table 3. We construct a binary classification problem by combining the major structural classes $\{\alpha, \beta\}$ into one class and $\{\alpha/\beta, \alpha+\beta\}$ into another class. Due to the small size of this data set, we use 30-fold cross-validation and the paired t test. We do three experiments on this data set using three different subsets of kernels.

Name	Dimension	Data Source
COM	20	Amino-acid composition
SEC	21	Predicted secondary structure
HYD	21	Hydrophobicity
VOL	21	Van der Waals volume
POL	21	Polarity
PLZ	21	Polarizability
L1	22	Pseudo amino-acid composition at interval 1
L4	28	Pseudo amino-acid composition at interval 4
L14	48	Pseudo amino-acid composition at interval 14
L30	80	Pseudo amino-acid composition at interval 30
BLO	311	Smith-Waterman scores with the BLOSUM 62 matrix
PAM	311	Smith-Waterman scores with the PAM 50 matrix

Table 3: Multiple feature representations in the PROTEIN data set.

Table 4 lists the performance values of all algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ). All combination algorithms except RBMKL (product) and GMKL outperform SVM (best) by more than four per cent in terms of average test accuracy. NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are the only four algorithms that obtain more than 80 per cent average test accuracy and are statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). Nonlinear combination algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), have the disadvantage that they store statistically significantly more

4. Available at `http://mkl.ucsd.edu/dataset/protein-fold-prediction`.

support vectors than all other algorithms. ABMKL (conic) and CABMKL (conic) are the two MKL algorithms that perform kernel selection and use less than five kernels on the average, while the others use all six kernels, except CABMKL (linear) which uses five kernels in one of 30 folds. The two-step algorithms, except GMKL, LMKL (softmax), and LMKL (sigmoid), need to solve fewer than 20 SVM problems on the average. GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms. LMKL (softmax) and LMKL (sigmoid) solve many SVM problems; the large standard deviations for this performance value are mainly due to the random initialization of the gating model parameters and it takes longer for some folds to converge.

Table 5 summarizes the performance values of all algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30). All combination algorithms except RBMKL (product) outperform SVM (best) by more than two per cent in terms of average test accuracy. NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithm that obtain more than 85 per cent average test accuracy and are statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). When the number of kernels combined becomes large as in this experiment, as a result of multiplication, RBMKL (product) starts to have very small kernel values at the off-diagonal entries of the combined kernel matrix. This causes the classifier to behave like a nearest-neighbor classifier by storing many support vectors and to perform badly in terms of average test accuracy. As observed in the previous experiment, the nonlinear combination algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than all other algorithms. ABMKL (conic), ABMKL (convex), CABMKL (linear), CABMKL (conic), MKL, SimpleMKL, and GMKL are the seven MKL algorithms that perform kernel selection and use fewer than 10 kernels on the average, while others use all 10 kernels. Similar to the results of the previous experiment, GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms and the very high standard deviations for LMKL (softmax) and LMKL (sigmoid) are also observed in this experiment.

Table 6 gives the performance values of all algorithms on the PROTEIN data set with a larger set of kernels, namely, (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30-BLO-PAM). All combination algorithms except RBMKL (product) outperform SVM (best) by more than three per cent in terms of average test accuracy. NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithms that obtain more than 87 per cent average test accuracy. In this experiment, ABMKL (ratio), GMKL, GLMKL ($p = 1$), GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) are statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). As noted in the two previous experiments, the nonlinear combination algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than all other algorithms. ABMKL (conic), ABMKL (convex), CABMKL (linear), CABMKL (conic), MKL, SimpleMKL, and GMKL are the seven MKL algorithms that perform kernel selection and use fewer than 12 kernels on the average, while others use all 12 kernels, except GLMKL ($p = 1$) which uses 11 kernels in one of 30 folds. Similar to the results of the two previous experiments, GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms, but the very high standard deviations for LMKL (softmax) and LMKL (sigmoid) are not observed in this experiment.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	72.06±0.74 $^{\underline{bc}}$	58.29±1.00 $^{\underline{bc}}$	1.00±0.00 $^{\underline{bc}}$	6.00± 0.00 $^{\overline{bc}}$
SVM (all)	79.13±0.45$^{\overline{a}\;\overline{c}}$	62.14±1.04$^{\overline{a}\;\overline{c}}$	6.00±0.00	1.00± 0.00
RBMKL (mean)	78.01±0.63	60.89±1.02	6.00±0.00	1.00± 0.00
RBMKL (product)	72.35±0.95 $^{\underline{bc}}$	100.00±0.00$^{\overline{abc}}$	6.00±0.00	1.00± 0.00
ABMKL (conic)	79.03±0.92$^{\overline{a}\;\overline{c}}$	49.96±1.01$^{\underline{abc}}$	4.60±0.50$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	76.90±1.17$^{\overline{a}\underline{bc}}$	29.54±0.89$^{\underline{abc}}$	6.00±0.00	1.00± 0.00
ABMKL (ratio)	78.06±0.62	56.95±1.07$^{\underline{abc}}$	6.00±0.00	1.00± 0.00
CABMKL (linear)	79.51±0.78$^{\overline{abc}}$	49.81±0.82$^{\underline{abc}}$	5.97±0.18$^{\overline{a}\underline{bc}}$	1.00± 0.00
CABMKL (conic)	79.28±0.97$^{\overline{a}\;\overline{c}}$	49.84±0.77$^{\underline{abc}}$	4.73±0.52$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	76.38±1.19$^{\overline{a}\underline{bc}}$	29.65±1.02$^{\underline{abc}}$	6.00±0.00	1.00± 0.00
SimpleMKL	76.34±1.24$^{\overline{a}\underline{bc}}$	29.62±1.08$^{\underline{abc}}$	6.00±0.00	18.83± 4.27$^{\overline{abc}}$
GMKL	74.96±0.50$^{\overline{a}\underline{bc}}$	79.85±0.70$^{\overline{abc}}$	2.37±0.56$^{\overline{a}\underline{bc}}$	37.10± 3.23$^{\overline{abc}}$
GLMKL ($p = 1$)	77.71±0.96	55.80±0.95$^{\underline{abc}}$	6.00±0.00	6.10± 0.31$^{\overline{abc}}$
GLMKL ($p = 2$)	77.20±0.42$^{\overline{a}\underline{bc}}$	75.34±0.70$^{\overline{abc}}$	6.00±0.00	5.00± 0.00$^{\underline{a}\overline{bc}}$
NLMKL ($p = 1$)	83.49±0.76$^{\overline{abc}}$	85.67±0.86$^{\overline{abc}}$	6.00±0.00	17.50± 0.51$^{\overline{abc}}$
NLMKL ($p = 2$)	82.30±0.62$^{\overline{abc}}$	89.57±0.77$^{\overline{abc}}$	6.00±0.00	13.40± 4.41$^{\overline{abc}}$
LMKL (softmax)	80.24±1.37$^{\overline{abc}}$	27.24±1.76$^{\underline{abc}}$	6.00±0.00	85.27±41.77$^{\overline{abc}}$
LMKL (sigmoid)	81.91±0.92$^{\overline{abc}}$	30.95±2.74$^{\underline{abc}}$	6.00±0.00	103.90±62.69$^{\overline{abc}}$

Table 4: Performances of single-kernel SVM and representative MKL algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	72.15±0.68 $^{\underline{bc}}$	47.50±1.25 $^{\overline{b}\underline{c}}$	1.00±0.00 $^{\underline{bc}}$	10.00± 0.00 $^{\overline{bc}}$
SVM (all)	79.63±0.74$^{\overline{a}\;\underline{c}}$	43.45±1.00$^{\underline{a}\;\underline{c}}$	10.00±0.00	1.00± 0.00
RBMKL (mean)	81.32±0.74	61.67±1.31	10.00±0.00	1.00± 0.00
RBMKL (product)	53.04±0.21$^{\underline{abc}}$	100.00±0.00$^{\overline{abc}}$	10.00±0.00	1.00± 0.00
ABMKL (conic)	80.45±0.68$^{\overline{ab}\underline{c}}$	48.16±1.08$^{\overline{ab}\underline{c}}$	6.90±0.66$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	77.47±0.62$^{\overline{a}\underline{bc}}$	87.86±0.76$^{\overline{abc}}$	9.03±0.61$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (ratio)	76.22±1.14$^{\overline{a}\underline{bc}}$	35.54±1.01$^{\underline{abc}}$	10.00±0.00	1.00± 0.00
CABMKL (linear)	77.15±0.63$^{\overline{a}\underline{bc}}$	73.84±0.80$^{\overline{abc}}$	9.90±0.31$^{\overline{a}\underline{bc}}$	1.00± 0.00
CABMKL (conic)	81.02±0.67	48.32±0.86$^{\overline{ab}\underline{c}}$	6.93±0.74$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	79.74±1.02$^{\overline{a}\;\underline{c}}$	56.00±0.85$^{\overline{ab}\underline{c}}$	8.73±0.52$^{\overline{a}\underline{bc}}$	1.00± 0.00
SimpleMKL	74.53±0.90$^{\overline{a}\underline{bc}}$	80.22±1.05$^{\overline{abc}}$	4.73±1.14$^{\overline{a}\underline{bc}}$	23.83± 7.46$^{\overline{abc}}$
GMKL	74.68±0.68$^{\overline{a}\underline{bc}}$	80.36±0.83$^{\overline{abc}}$	5.73±0.91$^{\overline{a}\underline{bc}}$	29.10± 8.47$^{\overline{abc}}$
GLMKL ($p = 1$)	79.77±0.86$^{\overline{a}\;\underline{c}}$	55.94±0.93$^{\overline{ab}\underline{c}}$	10.00±0.00	6.87± 0.57$^{\underline{a}\overline{bc}}$
GLMKL ($p = 2$)	78.00±0.43$^{\overline{a}\underline{bc}}$	72.49±1.00$^{\overline{abc}}$	10.00±0.00	5.03± 0.18$^{\underline{a}\overline{bc}}$
NLMKL ($p = 1$)	85.38±0.70$^{\overline{abc}}$	93.84±0.51$^{\overline{abc}}$	10.00±0.00	14.77± 0.43$^{\overline{abc}}$
NLMKL ($p = 2$)	85.40±0.69$^{\overline{abc}}$	93.86±0.51$^{\overline{abc}}$	10.00±0.00	18.00± 0.00$^{\overline{abc}}$
LMKL (softmax)	81.11±1.82	36.00±3.61$^{\underline{abc}}$	10.00±0.00	34.40±23.12$^{\overline{abc}}$
LMKL (sigmoid)	81.90±2.01	51.94±2.14$^{\overline{ab}\underline{c}}$	10.00±0.00	31.63±13.17$^{\overline{abc}}$

Table 5: Performances of single-kernel SVM and representative MKL algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	78.37±1.08 $^{\underline{bc}}$	93.09±0.73 $^{\overline{bc}}$	1.00±0.00 $^{\underline{bc}}$	12.00± 0.00 $^{\overline{bc}}$
SVM (all)	82.01±0.76$^{\overline{a}\ \underline{c}}$	89.32±0.99$^{\underline{a}\ \overline{c}}$	12.00±0.00	1.00± 0.00
RBMKL (mean)	83.57±0.59	65.94±0.93	12.00±0.00	1.00± 0.00
RBMKL (product)	53.04±0.21$^{\underline{abc}}$	100.00±0.00$^{\overline{abc}}$	12.00±0.00	1.00± 0.00
ABMKL (conic)	83.52±0.94	63.07±1.35$^{\underline{abc}}$	7.30±0.88$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	83.76±1.02	64.36±1.56$^{\underline{abc}}$	6.87±0.94$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (ratio)	85.65±0.67$^{\overline{abc}}$	57.87±1.24$^{\underline{abc}}$	12.00±0.00	1.00± 0.00
CABMKL (linear)	83.48±0.92	68.00±1.48$^{\underline{ab}\overline{c}}$	11.87±0.35$^{\overline{a}\underline{bc}}$	1.00± 0.00
CABMKL (conic)	83.43±0.95	62.12±1.63$^{\underline{abc}}$	8.43±0.73$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	83.55±1.25	81.75±1.06$^{\underline{ab}\overline{c}}$	7.67±0.76$^{\overline{a}\underline{bc}}$	1.00± 0.00
SimpleMKL	83.96±1.20	86.41±0.98$^{\underline{ab}\overline{c}}$	9.83±0.91$^{\overline{a}\underline{bc}}$	54.53± 9.92$^{\overline{abc}}$
GMKL	85.67±0.91$^{\overline{abc}}$	79.53±2.71$^{\underline{ab}\overline{c}}$	9.93±0.74$^{\overline{a}\underline{bc}}$	47.40±10.81$^{\overline{abc}}$
GLMKL ($p=1$)	85.96±0.96$^{\overline{abc}}$	79.06±1.04$^{\underline{ab}\overline{c}}$	11.97±0.18$^{\overline{a}\underline{bc}}$	14.77± 0.57$^{\overline{abc}}$
GLMKL ($p=2$)	85.02±1.20$^{\overline{abc}}$	62.06±1.02$^{\underline{abc}}$	12.00±0.00	5.60± 0.67$^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	87.00±0.66$^{\overline{abc}}$	96.78±0.32$^{\overline{abc}}$	12.00±0.00	4.83± 0.38$^{\underline{a}\overline{bc}}$
NLMKL ($p=2$)	87.28±0.65$^{\overline{abc}}$	96.64±0.32$^{\overline{abc}}$	12.00±0.00	17.77± 0.43$^{\overline{abc}}$
LMKL (softmax)	83.72±1.35	37.55±2.54$^{\underline{abc}}$	12.00±0.00	25.97± 5.75$^{\overline{abc}}$
LMKL (sigmoid)	85.06±0.83$^{\overline{abc}}$	48.99±1.59$^{\underline{abc}}$	12.00±0.00	25.40± 9.36$^{\overline{abc}}$

Table 6: Performances of single-kernel SVM and representative MKL algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30-BLO-PAM) using the linear kernel.

4.4 Pendigits Digit Recognition Experiments

We perform experiments on the Pendigits (PENDIGITS) digit recognition data set[5] from the MKL Repository, composed of four different feature representations for 10,992 instances (7,494 for training and 3,498 for testing). The properties of these feature representations are summarized in Table 7. Two binary classification problems are generated from the PENDIGITS data set: In the PENDIGITS-EO data set, we separate even digits from odd digits; in the PENDIGITS-SL data set, we separate small (‘0’ - ‘4’) digits from large (‘5’ - ‘9’) digits.

Name	Dimension	Data Source
DYN	16	8 successive pen points on two-dimensional coordinate system
STA4	16	4×4 image bitmap representation
STA8	64	8×8 image bitmap representation
STA16	256	16×16 image bitmap representation

Table 7: Multiple feature representations in the PENDIGITS data set.

Table 8 summarizes the performance values of all algorithms on the PENDIGITS-EO data set. We see that SVM (best) is outperformed (by more than three per cent) by all other algorithms in

5. Available at `http://mkl.ucsd.edu/dataset/pendigits`.

terms of average test accuracy, which implies that integrating different information sources helps. RBMKL (product), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) achieve statistically significantly higher average test accuracies than the other MKL algorithms. NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithms that get more than 99 percent average test accuracy and improve the average test accuracy of RBMKL (mean) statistically significantly, by nearly six per cent. When we look at the percentages of support vectors stored, we see that RBMKL (product) stores statistically significantly more support vectors than the other algorithms, whereas LMKL (softmax) and LMKL (sigmoid) store statistically significantly fewer support vectors. All combination algorithms except ABMKL (convex) use four kernels in all folds. All two-step algorithms except LMKL (softmax) and LMKL (sigmoid) need to solve less than 15 SVM optimization problems on the average. As observed before, LMKL (softmax) and LMKL (sigmoid) have very high standard deviations in the number of SVM optimization calls due to the random initialization of the gating model parameters; note that convergence may be slow at times, but the standard deviations of the test accuracy are small.

Table 9 lists the performance values of all algorithms on the PENDIGITS-SL data set. We again see that SVM (best) is outperformed (more than five per cent) by all other algorithms in terms of average test accuracy. RBMKL (product), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) achieve statistically significantly higher average test accuracies than the other MKL algorithms. Similar to the results on the PENDIGITS-EO data set, NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithms that get more than 99 percent average test accuracy by improving the average test accuracy of RBMKL (mean) nearly eight per cent for this experiment. As observed on the PENDIGITS-EO data set, we see that RBMKL (product) stores statistically significantly more support vectors than the other algorithms, whereas LMKL (softmax) and LMKL (sigmoid) store fewer support vectors. All combination algorithms except ABMKL (convex) use four kernels in all folds, whereas this latter uses exactly three kernels in all folds by eliminating STA8 representation. All two-step algorithms except LMKL (softmax) and LMKL (sigmoid) need to solve less than 20 SVM optimization problems on the average. GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer SVM problems than the other two-step algorithms.

4.5 Multiple Features Digit Recognition Experiments

We perform experiments on the Multiple Features (MULTIFEAT) digit recognition data set[6] from the UCI Machine Learning Repository, composed of six different feature representations for 2,000 handwritten numerals. The properties of these feature representations are summarized in Table 10. Two binary classification problems are generated from the MULTIFEAT data set: In the MULTIFEAT-EO data set, we separate even digits from odd digits; in the MULTIFEAT-SL data set, we separate small (‘0’ - ‘4’) digits from large (‘5’ - ‘9’) digits. We do two experiments on these data set using two different subsets of feature representations.

Table 11 gives the performance values of all algorithms on the MULTIFEAT-EO data set with (FOU-KAR-PIX-ZER). Though all algorithms except CABMKL (linear) have higher average test accuracies than SVM (best); only LMKL (sigmoid) is statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). Note that even though RBMKL (product) is not more accurate than SVM (all) or RBMKL (mean), nonlinear and data-dependent algorithms, namely, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid), are more accurate than these two

6. Available at `http://archive.ics.uci.edu/ml/datasets/Multiple+Features`.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$88.93\pm0.28^{\underline{bc}}$	$20.90\pm1.22^{\overline{c}}$	$1.00\pm0.00^{\underline{bc}}$	$4.00\pm\ 0.00^{\overline{bc}}$
SVM (all)	$92.12\pm0.42^{\overline{a}\,\underline{c}}$	$22.22\pm0.72^{\overline{c}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (mean)	93.34 ± 0.28	18.91 ± 0.67	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (product)	$98.46\pm0.16^{\overline{abc}}$	$51.08\pm0.48^{\overline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (conic)	93.40 ± 0.15	$17.52\pm0.73^{\underline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (convex)	93.53 ± 0.26	$13.83\pm0.75^{\underline{abc}}$	$3.90\pm0.32^{\overline{a}\underline{bc}}$	$1.00\pm\ 0.00$
ABMKL (ratio)	93.35 ± 0.20	18.89 ± 0.68	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (linear)	93.42 ± 0.16	$17.48\pm0.74^{\underline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (conic)	93.42 ± 0.16	$17.48\pm0.74^{\underline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
MKL	93.28 ± 0.29	$19.20\pm0.67^{\underline{b}\overline{c}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
SimpleMKL	93.29 ± 0.27	19.04 ± 0.71	4.00 ± 0.00	$8.70\pm\ 3.92^{\overline{abc}}$
GMKL	93.28 ± 0.26	19.08 ± 0.72	4.00 ± 0.00	$8.60\pm\ 3.66^{\overline{abc}}$
GLMKL ($p=1$)	93.34 ± 0.27	19.02 ± 0.73	4.00 ± 0.00	$3.20\pm\ 0.63^{\underline{a}\overline{bc}}$
GLMKL ($p=2$)	93.32 ± 0.25	$16.91\pm0.61^{\underline{abc}}$	4.00 ± 0.00	$3.80\pm\ 0.42^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	$99.36\pm0.08^{\overline{abc}}$	19.55 ± 0.48	4.00 ± 0.00	$11.60\pm\ 6.26^{\overline{abc}}$
NLMKL ($p=2$)	$99.38\pm0.07^{\overline{abc}}$	19.79 ± 0.52	4.00 ± 0.00	$10.90\pm\ 4.31^{\overline{abc}}$
LMKL (softmax)	$97.14\pm0.39^{\overline{abc}}$	$7.25\pm0.65^{\underline{abc}}$	4.00 ± 0.00	$97.70\pm55.48^{\overline{abc}}$
LMKL (sigmoid)	$97.80\pm0.20^{\overline{abc}}$	$11.71\pm0.71^{\underline{abc}}$	4.00 ± 0.00	$87.70\pm47.30^{\overline{abc}}$

Table 8: Performances of single-kernel SVM and representative MKL algorithms on the PENDIGITS-EO data set using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$84.44\pm0.49^{\underline{bc}}$	$39.31\pm0.77^{\overline{bc}}$	$1.00\pm0.00^{\underline{bc}}$	$4.00\pm\ 0.00^{\overline{bc}}$
SVM (all)	$89.48\pm0.67^{\overline{a}\,\underline{c}}$	$19.55\pm0.61^{\underline{a}\,\overline{c}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (mean)	91.11 ± 0.34	16.22 ± 0.59	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (product)	$98.37\pm0.11^{\overline{abc}}$	$60.28\pm0.69^{\overline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (conic)	90.97 ± 0.49	$20.93\pm0.46^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (convex)	90.85 ± 0.51	$24.59\pm0.69^{\underline{a}\overline{bc}}$	$3.00\pm0.00^{\overline{a}\underline{bc}}$	$1.00\pm\ 0.00$
ABMKL (ratio)	91.12 ± 0.32	16.23 ± 0.57	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (linear)	91.02 ± 0.47	$20.89\pm0.49^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (conic)	91.02 ± 0.47	$20.90\pm0.50^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
MKL	90.85 ± 0.45	$23.59\pm0.56^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
SimpleMKL	90.84 ± 0.50	$23.48\pm0.55^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$14.50\pm\ 3.92^{\overline{abc}}$
GMKL	90.85 ± 0.47	$23.46\pm0.54^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$15.60\pm\ 3.34^{\overline{abc}}$
GLMKL ($p=1$)	90.90 ± 0.46	$23.33\pm0.57^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$4.90\pm\ 0.57^{\overline{abc}}$
GLMKL ($p=2$)	91.12 ± 0.44	$20.40\pm0.55^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$4.00\pm\ 0.00^{\overline{bc}}$
NLMKL ($p=1$)	$99.11\pm0.10^{\overline{abc}}$	17.37 ± 0.17	4.00 ± 0.00	$18.10\pm\ 0.32^{\overline{abc}}$
NLMKL ($p=2$)	$99.07\pm0.12^{\overline{abc}}$	17.66 ± 0.23	4.00 ± 0.00	$10.90\pm\ 3.70^{\overline{abc}}$
LMKL (softmax)	$97.77\pm0.54^{\overline{abc}}$	$5.72\pm0.46^{\underline{abc}}$	4.00 ± 0.00	$116.60\pm73.34^{\overline{abc}}$
LMKL (sigmoid)	$97.13\pm0.40^{\overline{abc}}$	$6.69\pm0.27^{\underline{abc}}$	4.00 ± 0.00	$119.00\pm45.04^{\overline{abc}}$

Table 9: Performances of single-kernel SVM and representative MKL algorithms on the PENDIGITS-SL data set using the linear kernel.

Name	Dimension	Data Source
FAC	216	Profile correlations
FOU	76	Fourier coefficients of the shapes
KAR	64	Karhunen-Loève coefficients
MOR	6	Morphological features
PIX	240	Pixel averages in 2×3 windows
ZER	47	Zernike moments

Table 10: Multiple feature representations in the MULTIFEAT data set.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$95.96 \pm 0.50\ ^{\underline{bc}}$	$21.37 \pm 0.81\ ^{\underline{c}}$	$1.00 \pm 0.00\ ^{\underline{bc}}$	$4.00 \pm 0.00\ ^{\overline{bc}}$
SVM (all)	97.79 ± 0.25	$21.63 \pm 0.73\ ^{\underline{c}}$	4.00 ± 0.00	1.00 ± 0.00
RBMKL (mean)	97.94 ± 0.29	23.42 ± 0.79	4.00 ± 0.00	1.00 ± 0.00
RBMKL (product)	$96.43 \pm 0.38\ ^{\underline{bc}}$	$92.11 \pm 1.18^{\overline{abc}}$	4.00 ± 0.00	1.00 ± 0.00
ABMKL (conic)	97.85 ± 0.25	$19.40 \pm 1.02^{\underline{abc}}$	$2.00 \pm 0.00^{\overline{a}\underline{bc}}$	1.00 ± 0.00
ABMKL (convex)	$95.97 \pm 0.57\ ^{\underline{bc}}$	$21.45 \pm 0.92\ ^{\underline{c}}$	$1.20 \pm 0.42\ ^{\underline{bc}}$	1.00 ± 0.00
ABMKL (ratio)	97.82 ± 0.32	$22.33 \pm 0.57\ ^{\overline{b}\underline{c}}$	4.00 ± 0.00	1.00 ± 0.00
CABMKL (linear)	$95.78 \pm 0.37\ ^{\underline{bc}}$	$19.25 \pm 1.09\ ^{\underline{bc}}$	4.00 ± 0.00	1.00 ± 0.00
CABMKL (conic)	97.85 ± 0.25	$19.37 \pm 1.03^{\underline{abc}}$	$2.00 \pm 0.00^{\overline{a}\underline{bc}}$	1.00 ± 0.00
MKL	97.88 ± 0.31	$21.01 \pm 0.87\ ^{\underline{c}}$	$3.50 \pm 0.53^{\overline{a}\underline{bc}}$	1.00 ± 0.00
SimpleMKL	97.87 ± 0.32	$20.90 \pm 0.94\ ^{\underline{c}}$	$3.40 \pm 0.70^{\overline{a}\underline{bc}}$	$22.50 \pm 6.65^{\overline{abc}}$
GMKL	97.88 ± 0.31	$21.00 \pm 0.88\ ^{\underline{c}}$	$3.50 \pm 0.53^{\overline{a}\underline{bc}}$	$25.90 \pm 10.05^{\overline{abc}}$
GLMKL ($p = 1$)	97.90 ± 0.25	$21.31 \pm 0.78\ ^{\underline{c}}$	4.00 ± 0.00	$11.10 \pm 0.74^{\overline{abc}}$
GLMKL ($p = 2$)	98.01 ± 0.24	$19.19 \pm 0.61\ ^{\underline{bc}}$	4.00 ± 0.00	$4.90 \pm 0.32^{\overline{abc}}$
NLMKL ($p = 1$)	98.67 ± 0.22	$56.91 \pm 1.17^{\overline{abc}}$	4.00 ± 0.00	$4.50 \pm 1.84\ ^{\overline{bc}}$
NLMKL ($p = 2$)	98.61 ± 0.24	$53.61 \pm 1.20^{\overline{abc}}$	4.00 ± 0.00	$5.60 \pm 3.03\ ^{\overline{bc}}$
LMKL (softmax)	98.16 ± 0.50	$17.40 \pm 1.17^{\underline{abc}}$	4.00 ± 0.00	$36.70 \pm 14.11^{\overline{abc}}$
LMKL (sigmoid)	$98.94 \pm 0.29^{\overline{abc}}$	$15.23 \pm 1.08^{\underline{abc}}$	4.00 ± 0.00	$88.20 \pm 36.00^{\overline{abc}}$

Table 11: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-EO data set with (FOU-KAR-PIX-ZER) using the linear kernel.

algorithms. Alignment-based and centered-alignment-based MKL algorithms, namely, ABMKL (ratio), ABMKL (conic), ABMKL (convex), CABMKL (linear) and CABMKL (convex), are not more accurate than RBMKL (mean). We see that ABMKL (convex) and CABMKL (linear) are statistically significantly less accurate than SVM (all) and RBMKL (mean). If we compare the algorithms in terms of support vector percentages, we note that MKL algorithms that use products of the combined kernels, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than all other algorithms. If we look at the active kernel counts, 10 out of 16 MKL algorithms use all four kernels. The two-step algorithms solve statistically significantly more optimization problems than the one-step algorithms.

Table 12 summarizes the performance values of all algorithms on the MULTIFEAT-EO data set with (FAC-FOU-KAR-MOR-PIX-ZER). We note that NLMKL ($p = 1$) and LMKL (sigmoid) are the two MKL algorithms that achieve average test accuracy greater than or equal to 99 per cent, while NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) are statistically significantly more accurate than RBMKL (mean). All other MKL algorithms except RBMKL (product) and CABMKL (linear) achieve average test accuracies between 98 per cent and 99 per cent. Similar to the results of the previous experiment, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$) store statistically significantly more support vectors than all other algorithms. When we look at the number of active kernels, ABMKL (convex) selects only one kernel and this is the same kernel that SVM (best) picks. ABMKL (conic) and CABMKL (conic) use three kernels, whereas all other algorithms use more than five kernels on the average. GLMKL ($p = 1$), GLMKL ($p = 2$), NLMKL ($p = 1$), and NLMKL ($p = 2$) solve fewer optimization problems than the other two-step algorithms, namely, SimpleMKL, GMKL, LMKL (softmax), and LMKL (sigmoid).

Table 13 lists the performance values of all algorithms on the MULTIFEAT-SL data set with (FOU-KAR-PIX-ZER). SVM (best) is outperformed by the other algorithms on the average and this shows that, for this data set, combining multiple information sources, independently of the combination algorithm used, improves the average test accuracy. RBMKL (product), NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) are the four MKL algorithms that achieve statistically significantly higher average test accuracies than RBMKL (best), SVM (all), RBMKL (mean). NLMKL ($p = 1$) and NLMKL ($p = 2$) are the two best algorithms and are statistically significantly more accurate than all other algorithms, except LMKL (sigmoid). However, NLMKL ($p = 1$) and NLMKL ($p = 2$) store statistically significantly more support vectors than all other algorithms, except RBMKL (product). All MKL algorithms use all of the kernels and the two-step algorithms solve statistically significantly more optimization problems than the one-step algorithms.

Table 14 gives the performance values of all algorithms on the MULTIFEAT-SL data set with (FAC-FOU-KAR-MOR-PIX-ZER). GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are the five MKL algorithms that achieve higher average test accuracies than RBMKL (mean). CABMKL (linear) is the only algorithm that has statistically significantly lower average test accuracy than SVM (best). No MKL algorithm achieves statistically significantly higher average test accuracies than SVM (best), SVM (all), and RBMKL (mean). MKL algorithms with nonlinear combination rules, namely, RBMKL (product), NLMKL ($p = 1$) and NLMKL ($p = 2$), again use more support vectors than the other algorithms, whereas LMKL with a data-dependent combination approach stores statistically significantly fewer support vectors. ABMKL (conic), ABMKL (convex), and CABMKL (conic) are the three MKL algorithms that perform kernel selection and use fewer than five kernels on the average, while others use all of the kernels. GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms and the very high standard deviations for LMKL (softmax) and LMKL (sigmoid) are also observed in this experiment.

4.6 Internet Advertisements Experiments

We perform experiments on the Internet Advertisements (ADVERT) data set[7] from the UCI Machine Learning Repository, composed of five different feature representations (different bags of words); there is also some additional geometry information of the images, but we ignore them in our experiments due to missing values. After removing the data instances with missing values, we have a total

7. Available at `http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements`.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	98.39 ± 0.36	$10.30\pm0.83^{\underline{bc}}$	$1.00\pm0.00^{\underline{bc}}$	$6.00\pm0.00^{\overline{bc}}$
SVM (all)	98.24 ± 0.40	14.44 ± 0.74	6.00 ± 0.00	1.00 ± 0.00
RBMKL (mean)	98.09 ± 0.31	15.16 ± 0.83	6.00 ± 0.00	1.00 ± 0.00
RBMKL (product)	$95.87\pm0.31^{\underline{abc}}$	$100.00\pm0.00^{\overline{abc}}$	6.00 ± 0.00	1.00 ± 0.00
ABMKL (conic)	98.24 ± 0.38	13.08 ± 0.93	$3.00\pm0.00^{\overline{a}\underline{bc}}$	1.00 ± 0.00
ABMKL (convex)	98.39 ± 0.36	$10.30\pm0.83^{\underline{bc}}$	$1.00\pm0.00^{\underline{bc}}$	1.00 ± 0.00
ABMKL (ratio)	98.19 ± 0.25	14.11 ± 0.64	6.00 ± 0.00	1.00 ± 0.00
CABMKL (linear)	96.90 ± 0.34	$16.89\pm0.91^{\overline{abc}}$	$5.90\pm0.32^{\overline{a}\underline{bc}}$	1.00 ± 0.00
CABMKL (conic)	98.15 ± 0.41	12.54 ± 0.75	$3.00\pm0.00^{\overline{a}\underline{bc}}$	1.00 ± 0.00
MKL	98.31 ± 0.34	14.88 ± 0.81	$5.40\pm0.70^{\overline{a}\underline{bc}}$	1.00 ± 0.00
SimpleMKL	98.25 ± 0.37	14.89 ± 0.70	$5.60\pm0.52^{\overline{a}\underline{bc}}$	$37.50\pm12.09^{\overline{abc}}$
GMKL	98.24 ± 0.34	$14.33\pm0.85^{\overline{a}\ \underline{c}}$	$5.60\pm0.52^{\overline{a}\underline{bc}}$	$31.70\pm10.79^{\overline{abc}}$
GLMKL ($p=1$)	98.28 ± 0.31	$14.44\pm0.87^{\overline{a}\ \underline{c}}$	6.00 ± 0.00	$9.30\pm1.25^{\overline{abc}}$
GLMKL ($p=2$)	98.37 ± 0.28	$17.04\pm0.80^{\overline{abc}}$	6.00 ± 0.00	$4.90\pm0.32^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	$99.00\pm0.16^{\overline{c}}$	$47.50\pm1.27^{\overline{abc}}$	6.00 ± 0.00	$8.30\pm2.71^{\overline{abc}}$
NLMKL ($p=2$)	$98.93\pm0.18^{\overline{c}}$	$46.78\pm1.07^{\overline{abc}}$	6.00 ± 0.00	$12.00\pm3.16^{\overline{abc}}$
LMKL (softmax)	98.34 ± 0.25	11.36 ± 1.83	6.00 ± 0.00	$94.90\pm24.73^{\overline{abc}}$
LMKL (sigmoid)	$99.24\pm0.18^{\overline{c}}$	17.88 ± 1.06	6.00 ± 0.00	$94.90\pm57.64^{\overline{abc}}$

Table 12: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-EO data set with (FAC-FOU-KAR-MOR-PIX-ZER) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$90.54\pm1.12^{\underline{bc}}$	$28.90\pm1.69^{\underline{b}\overline{c}}$	$1.00\pm0.00^{\underline{bc}}$	$4.00\pm0.00^{\overline{bc}}$
SVM (all)	94.45 ± 0.44	$40.26\pm1.28^{\overline{a}\ \overline{c}}$	4.00 ± 0.00	1.00 ± 0.00
RBMKL (mean)	95.00 ± 0.76	24.73 ± 1.19	4.00 ± 0.00	1.00 ± 0.00
RBMKL (product)	$96.51\pm0.31^{\overline{abc}}$	$95.31\pm0.60^{\overline{abc}}$	4.00 ± 0.00	1.00 ± 0.00
ABMKL (conic)	95.12 ± 0.36	$33.44\pm1.20^{\overline{a}\underline{b}\overline{c}}$	4.00 ± 0.00	1.00 ± 0.00
ABMKL (convex)	94.51 ± 0.59	24.34 ± 1.19	4.00 ± 0.00	1.00 ± 0.00
ABMKL (ratio)	94.93 ± 0.73	24.88 ± 1.02	4.00 ± 0.00	1.00 ± 0.00
CABMKL (linear)	95.10 ± 0.38	$33.44\pm1.24^{\overline{a}\underline{b}\overline{c}}$	4.00 ± 0.00	1.00 ± 0.00
CABMKL (conic)	95.10 ± 0.38	$33.44\pm1.24^{\overline{a}\underline{b}\overline{c}}$	4.00 ± 0.00	1.00 ± 0.00
MKL	94.81 ± 0.67	24.46 ± 1.13	4.00 ± 0.00	1.00 ± 0.00
SimpleMKL	94.84 ± 0.64	24.40 ± 1.18	4.00 ± 0.00	$15.50\pm8.11^{\overline{abc}}$
GMKL	94.84 ± 0.64	24.41 ± 1.18	4.00 ± 0.00	$15.60\pm8.07^{\overline{abc}}$
GLMKL ($p=1$)	94.84 ± 0.69	24.34 ± 1.27	4.00 ± 0.00	$6.20\pm1.03^{\overline{abc}}$
GLMKL ($p=2$)	95.18 ± 0.32	$32.34\pm1.36^{\overline{a}\underline{b}\overline{c}}$	4.00 ± 0.00	$4.20\pm0.63^{\overline{bc}}$
NLMKL ($p=1$)	$98.64\pm0.25^{\overline{abc}}$	$50.17\pm1.31^{\overline{abc}}$	4.00 ± 0.00	$9.20\pm4.80^{\overline{abc}}$
NLMKL ($p=2$)	$98.63\pm0.28^{\overline{abc}}$	$57.02\pm1.26^{\overline{abc}}$	4.00 ± 0.00	$9.10\pm3.28^{\overline{abc}}$
LMKL (softmax)	96.24 ± 0.90	24.16 ± 3.29	4.00 ± 0.00	$41.70\pm31.28^{\overline{abc}}$
LMKL (sigmoid)	$97.16\pm0.60^{\overline{abc}}$	$20.18\pm1.06^{\underline{abc}}$	4.00 ± 0.00	$75.50\pm28.38^{\overline{abc}}$

Table 13: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-SL data set with (FOU-KAR-PIX-ZER) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$94.99 \pm 0.85\ ^{\underline{bc}}$	$17.96 \pm 0.89\ ^{\underline{bc}}$	$1.00 \pm 0.00\ ^{\underline{bc}}$	$6.00 \pm 0.00\ ^{\overline{bc}}$
SVM (all)	97.69 ± 0.44	23.34 ± 1.13	6.00 ± 0.00	1.00 ± 0.00
RBMKL (mean)	97.67 ± 0.50	20.98 ± 0.84	6.00 ± 0.00	1.00 ± 0.00
RBMKL (product)	$96.01 \pm 0.17\ ^{\underline{bc}}$	$97.58 \pm 0.48^{\overline{abc}}$	6.00 ± 0.00	1.00 ± 0.00
ABMKL (conic)	96.84 ± 0.39	$27.49 \pm 0.92^{\overline{abc}}$	$4.50 \pm 0.53^{\overline{a}\underline{bc}}$	1.00 ± 0.00
ABMKL (convex)	96.46 ± 0.34	$33.78 \pm 0.90^{\overline{abc}}$	$4.60 \pm 0.52^{\overline{a}\underline{bc}}$	1.00 ± 0.00
ABMKL (ratio)	97.66 ± 0.46	20.95 ± 0.88	6.00 ± 0.00	1.00 ± 0.00
CABMKL (linear)	$89.18 \pm 0.81^{\underline{abc}}$	$57.22 \pm 1.47^{\overline{abc}}$	6.00 ± 0.00	1.00 ± 0.00
CABMKL (conic)	96.84 ± 0.39	$27.57 \pm 0.95^{\overline{abc}}$	$4.50 \pm 0.53^{\overline{a}\underline{bc}}$	1.00 ± 0.00
MKL	97.40 ± 0.37	$32.59 \pm 0.82^{\overline{abc}}$	6.00 ± 0.00	1.00 ± 0.00
SimpleMKL	97.51 ± 0.37	$32.53 \pm 0.94^{\overline{abc}}$	6.00 ± 0.00	$14.40 \pm 3.27^{\overline{abc}}$
GMKL	97.51 ± 0.35	$32.73 \pm 1.01^{\overline{abc}}$	6.00 ± 0.00	$14.20 \pm 4.59^{\overline{abc}}$
GLMKL ($p = 1$)	97.51 ± 0.28	$32.49 \pm 0.93^{\overline{abc}}$	6.00 ± 0.00	$6.70 \pm 0.95\ ^{\overline{bc}}$
GLMKL ($p = 2$)	97.81 ± 0.22	$25.19 \pm 1.06^{\overline{abc}}$	6.00 ± 0.00	$5.00 \pm 0.82^{\underline{a}\overline{bc}}$
NLMKL ($p = 1$)	98.79 ± 0.28	$38.44 \pm 0.96^{\overline{abc}}$	6.00 ± 0.00	$12.10 \pm 3.98^{\overline{abc}}$
NLMKL ($p = 2$)	98.82 ± 0.20	$43.99 \pm 0.99^{\overline{abc}}$	6.00 ± 0.00	$10.70 \pm 4.62^{\overline{abc}}$
LMKL (softmax)	97.79 ± 0.62	$14.71 \pm 1.10\ ^{\underline{bc}}$	6.00 ± 0.00	$59.00 \pm 31.42^{\overline{abc}}$
LMKL (sigmoid)	98.48 ± 0.70	$16.10 \pm 2.09\ ^{\underline{bc}}$	6.00 ± 0.00	$107.60 \pm 76.90^{\overline{abc}}$

Table 14: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-SL data set with (FAC-FOU-KAR-MOR-PIX-ZER) using the linear kernel.

of 3,279 images in the data set. The properties of these feature representations are summarized in Table 15. The classification task is to predict whether an image is an advertisement or not.

Name	Dimension	Data Source
URL	457	Phrases occurring in the URL
ORIGURL	495	Phrases occurring in the URL of the image
ANCURL	472	Phrases occurring in the anchor text
ALT	111	Phrases occurring in the alternative text
CAPTION	19	Phrases occurring in the caption terms

Table 15: Multiple feature representations in the ADVERT data set.

Table 16 lists the performance values of all algorithms on the ADVERT data set. We can see that all MKL algorithms except RBMKL (product) achieve similar average test accuracies. However, no MKL algorithm is statistically significantly more accurate than RBMKL (mean), and ABMKL (convex) is statistically significantly worse. We see again that algorithms that combine kernels by multiplying them, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than other MKL algorithms. 10 out of 16 MKL algorithms use all five kernels; ABMKL (conic) and ABMKL (convex) eliminate two representations, namely, URL and ORIGURL. GMKL $(p = 1)$ and GMKL ($p = 2$) solve statistically significantly fewer optimization problems than the other two-step algorithms.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	95.45±0.31	64.90± 5.41 $^{\overline{bc}}$	1.00±0.00 $^{\underline{bc}}$	5.00± 0.00 $^{\overline{bc}}$
SVM (all)	96.43±0.24	41.99± 1.76	5.00±0.00	1.00± 0.00
RBMKL (mean)	96.53±0.58	34.40± 4.25	5.00±0.00	1.00± 0.00
RBMKL (product)	89.98±0.49$^{\underline{abc}}$	96.61± 1.71$^{\overline{abc}}$	5.00±0.00	1.00± 0.00
ABMKL (conic)	95.69±0.27	44.16± 2.65$^{\underline{a}\ \overline{c}}$	3.00±0.00$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	95.10±0.52 $^{\underline{bc}}$	58.07± 2.47 $^{\overline{bc}}$	3.00±0.00$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (ratio)	96.23±0.61	35.07± 2.92	5.00±0.00	1.00± 0.00
CABMKL (linear)	95.86±0.19	36.43± 1.50	5.00±0.00	1.00± 0.00
CABMKL (conic)	95.84±0.19	38.06± 2.36	4.40±0.52$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	96.32±0.50	35.82± 4.35	4.10±0.32$^{\overline{a}\underline{bc}}$	1.00± 0.00
SimpleMKL	96.37±0.46	33.78± 4.40	4.60±0.52$^{\overline{a}\underline{bc}}$	27.00± 7.39$^{\overline{abc}}$
GMKL	96.40±0.49	33.18± 3.49	4.70±0.48$^{\overline{a}\underline{bc}}$	27.20± 7.94$^{\overline{abc}}$
GLMKL ($p = 1$)	96.35±0.55	32.81± 3.56	5.00±0.00	5.40± 1.07 $^{\overline{bc}}$
GLMKL ($p = 2$)	96.56±0.32	35.62± 1.55	5.00±0.00	4.90± 0.74 $^{\overline{bc}}$
NLMKL ($p = 1$)	95.96±0.50	67.63± 3.46 $^{\overline{bc}}$	5.00±0.00	15.90± 5.38$^{\overline{abc}}$
NLMKL ($p = 2$)	96.13±0.31	65.70± 3.03 $^{\overline{bc}}$	5.00±0.00	13.00± 0.00$^{\overline{abc}}$
LMKL (softmax)	95.68±0.53	24.18± 5.74	5.00±0.00	38.80±24.11$^{\overline{abc}}$
LMKL (sigmoid)	95.49±0.48	18.22±12.16	5.00±0.00	56.60±53.70$^{\overline{abc}}$

Table 16: Performances of single-kernel SVM and representative MKL algorithms on the ADVERT data set using the linear kernel.

4.7 Overall Comparison

After comparing algorithms for each experiment separately, we give an overall comparison on 10 experiments using the nonparametric Friedman's test on rankings with the Tukey's honestly significant difference criterion as the post-hoc test (Demšar, 2006).

Figure 2 shows the overall comparison between the algorithms in terms of misclassification error. First of all, we see that combining multiple information sources clearly improves the classification performance because SVM (best) is worse than all other algorithms. GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are statistically significantly more accurate than SVM (best). MKL algorithms using a trained, weighted combination on the average seem a little worse (but not statistically significantly) than the untrained, unweighted sum, namely, RBMKL (mean). NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are more accurate (but not statistically significantly) than RBMKL (mean). These results seem to suggest that if we want to improve the classification accuracy of MKL algorithms, we should investigate nonlinear and data-dependent approaches to better exploit information provided by different kernels.

Figure 3 illustrates the overall comparison between the algorithms in terms of the support vector percentages. We note that algorithms are clustered into three groups: (a) nonlinear MKL algorithms, (b) single-kernel SVM and linear MKL algorithms, and (c) data-dependent MKL algorithms. Nonlinear MKL algorithms, namely, RBMKL (product), NLMKL ($p = 1$) and NLMKL ($p = 2$), store more (but not statistically significantly) support vectors than single-kernel SVM and linear MKL algorithms, whereas they store statistically significantly more support vectors than

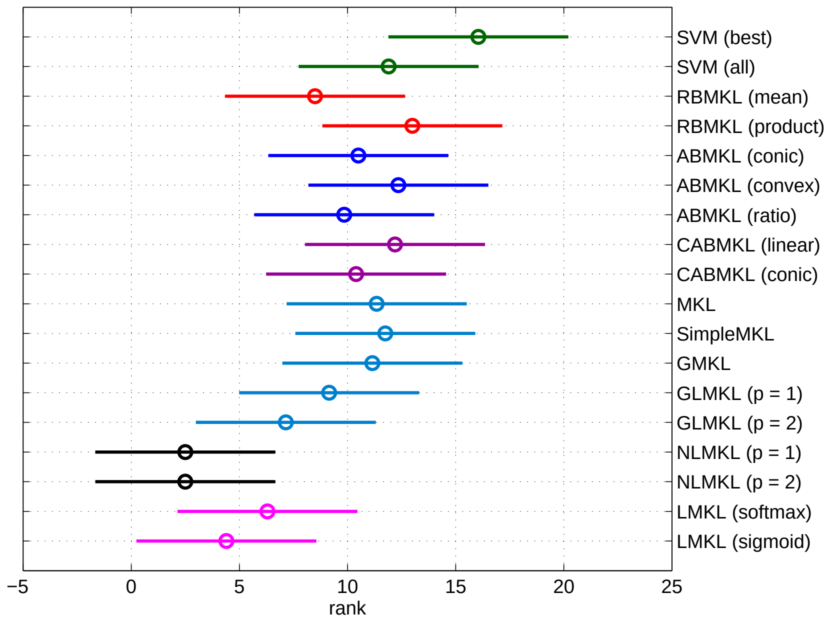

Figure 2: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of misclassification error using the linear kernel.

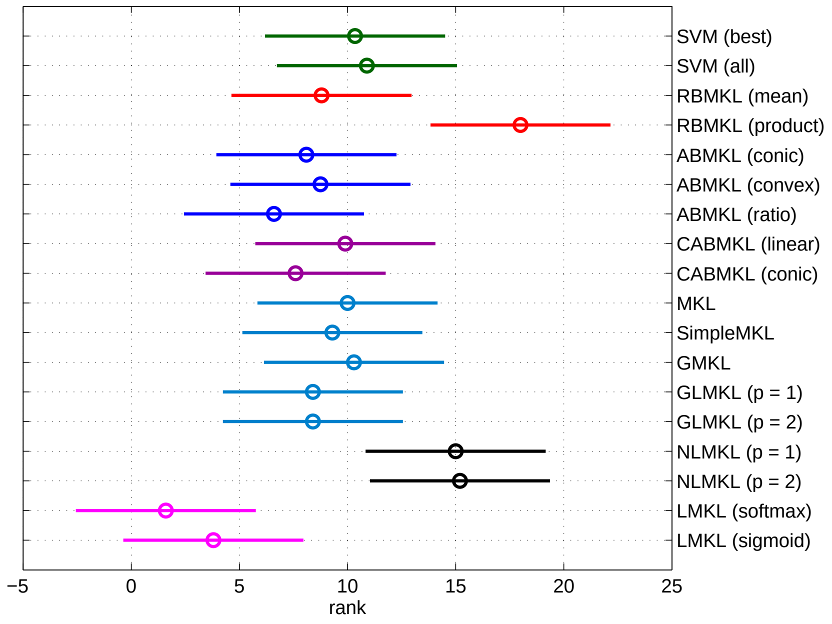

Figure 3: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of support vector percentages using the linear kernel.

data-dependent MKL algorithms. Data-dependent MKL algorithms, namely, LMKL (softmax) and LMKL (sigmoid), store fewer (but not statistically significantly) support vectors than single-kernel SVM and linear MKL algorithms, whereas LMKL (softmax) stores statistically significantly fewer support vectors than SVM (best) and SVM (all).

Figure 4 gives the overall comparison between the algorithms in terms of active kernel counts. We see that ABMKL (conic), ABMKL (convex), CABMKL (linear), CABMKL (conic), MKL, SimpleMKL, and GMKL use fewer kernels (statistically significantly in the case of the first two algorithms) than other combination algorithms. Even if we optimize the alignment and centered-alignment measures without any regularization on kernel weights using ABMKL (conic), ABMKL (convex), and CABMKL (conic), we obtain more sparse (but not statistically significantly) kernel combinations than MKL and SimpleMKL, which regularize kernel weights using the ℓ_1-norm. Trained nonlinear and data-dependent MKL algorithms, namely, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid), tend to use all of the kernels without eliminating any of them, whereas data-dependent algorithms use the kernels in different parts of the feature space with the help of the gating model.

Figure 5 shows the overall comparison between the algorithms in terms of the optimization toolbox call counts. We clearly see that the two-step algorithms need to solve more optimization problems than the other combination algorithms. SimpleMKL, GMKL, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) require solving statistically significantly more optimization problems than the one-step algorithms, whereas the differences between the one-step algorithms and GLMKL ($p = 1$) and GLMKL ($p = 2$) are not statistically significant.

4.8 Overall Comparison Using Gaussian Kernel

We also replicate the same set of experiments, except on PENDIGITS data set, using three different Gaussian kernels for each feature representation. We select the kernel widths as $\{\sqrt{D_m}/2, \sqrt{D_m}, 2\sqrt{D_m}\}$ where D_m is the dimensionality of the corresponding feature representation.

Figure 6 shows the overall comparison between the algorithms in terms of misclassification error. We see that no MKL algorithm is statistically significantly better than RBMKL (mean) and conclude that combining complex Gaussian kernels does not help much. ABMKL (ratio), MKL, SimpleMKL, GMKL, GLMKL ($p = 1$), and GLMKL ($p = 2$) obtain accuracy results comparable to RBMKL (mean). As an important result, we see that nonlinear and data-dependent MKL algorithms, namely, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid), are outperformed (but not statistically significantly) by RBMKL (mean). If we have highly nonlinear kernels such as Gaussian kernels, there is no need to combine them in a nonlinear or data-dependent way.

Figure 7 illustrates the overall comparison between the algorithms in terms of the support vector percentages. Different from the results obtained with simple linear kernels, algorithms do not exhibit a clear grouping. However, data-dependent MKL algorithms, namely, LMKL (softmax) and LMKL (sigmoid), tend to use fewer support vectors, whereas nonlinear MKL algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), tend to store more support vectors than other algorithms.

Figure 8 gives the overall comparison between the algorithms in terms of active kernel counts. ABMKL (ratio), GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) do not eliminate any of the base kernels even though we have three different kernels for each feature representation. When combining complex Gaussian kernels, trained MKL algorithms do not improve the classification performance statistically significantly, but they can eliminate some of the kernels.

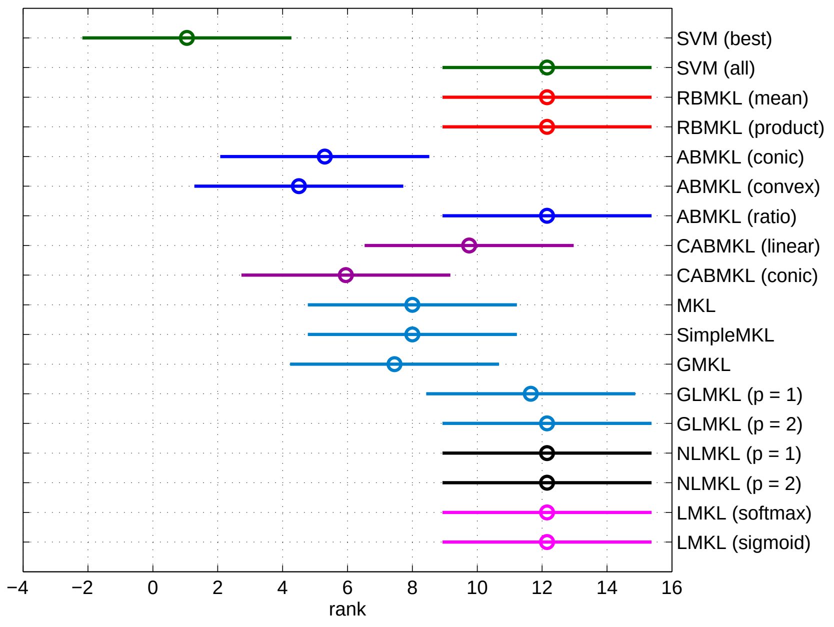

Figure 4: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of active kernel counts using the linear kernel.

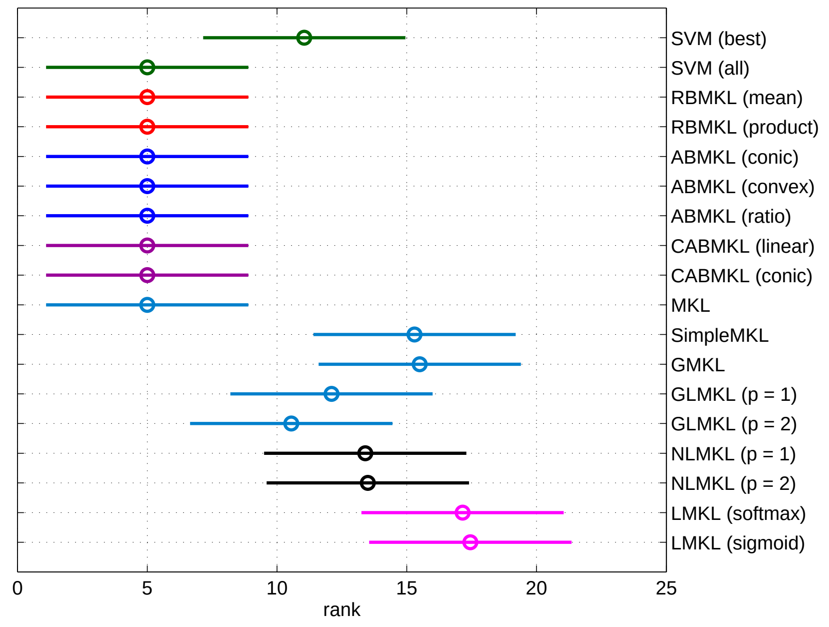

Figure 5: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of optimization toolbox call counts using the linear kernel.

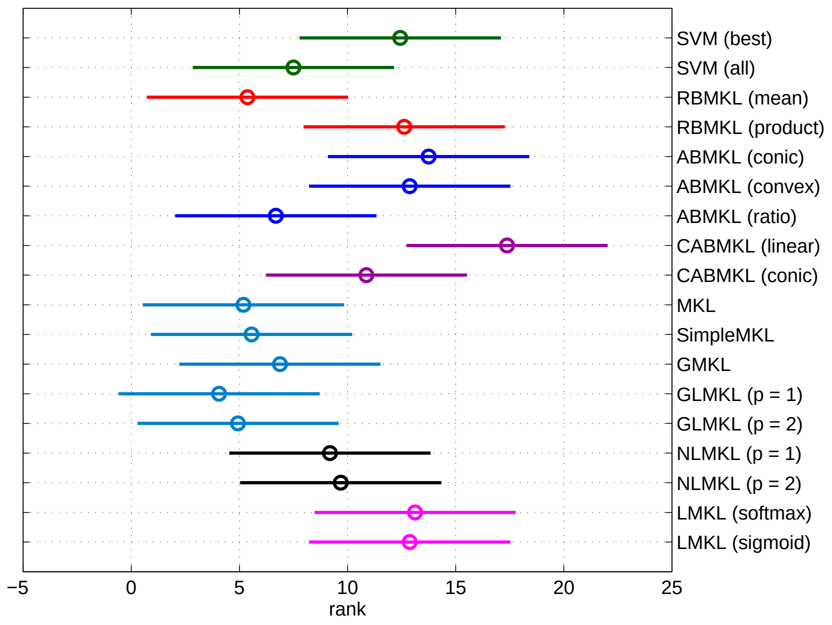

Figure 6: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of misclassification error using the Gaussian kernel.

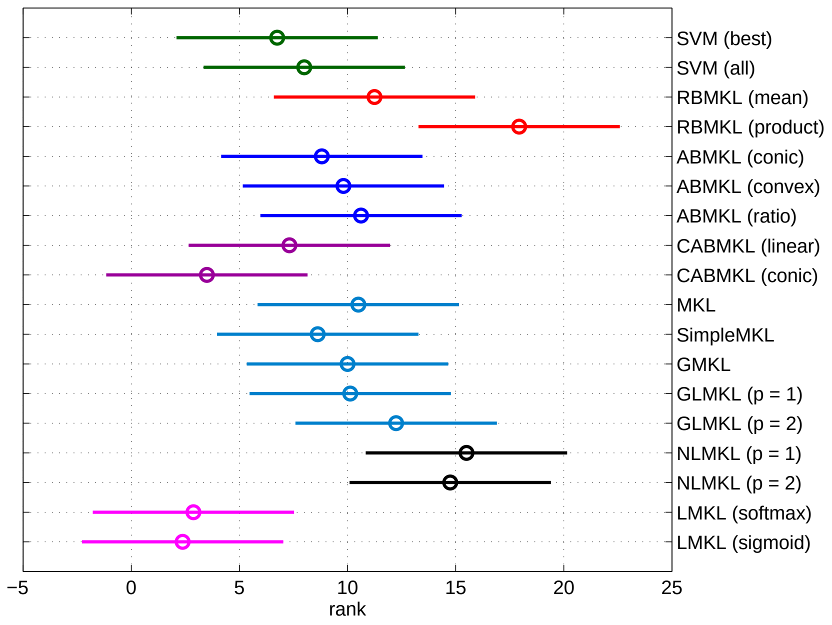

Figure 7: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of support vector percentages using the Gaussian kernel.

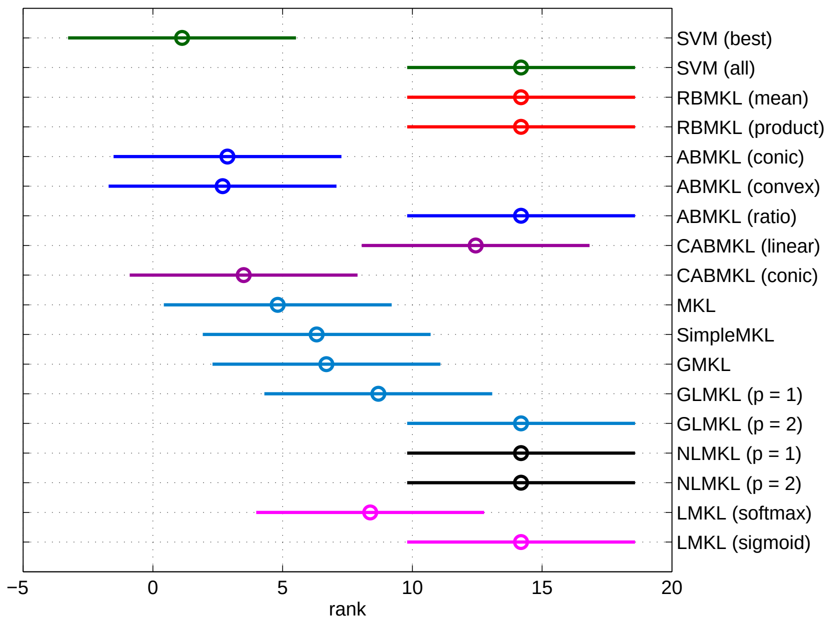

Figure 8: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of active kernel counts using the Gaussian kernel.

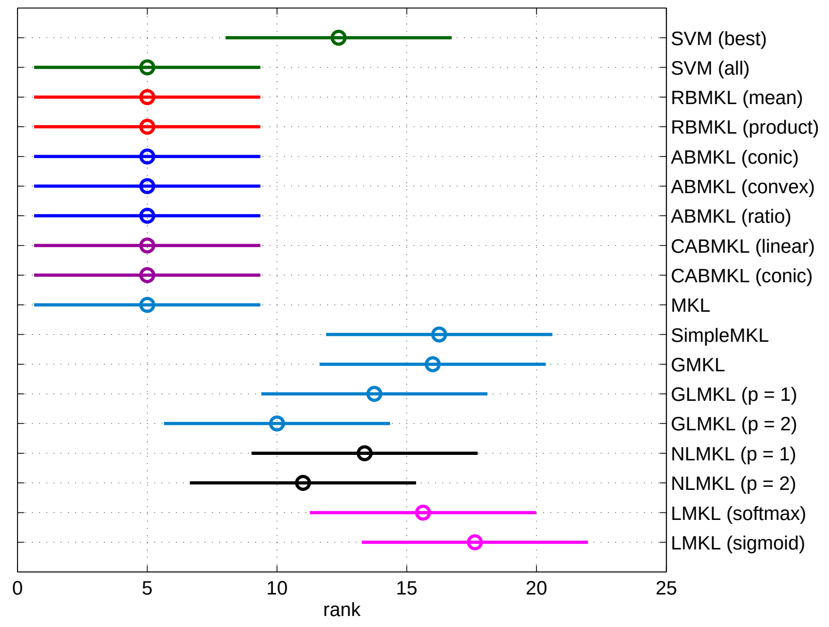

Figure 9: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of optimization toolbox call counts using the Gaussian kernel.

We see that ABMKL (conic), ABMKL (convex), CABMKL (conic), MKL, SimpleMKL, GMKL, GLMKL ($p = 1$), and LMKL (softmax) use fewer kernels (statistically significantly in the case of the first three algorithms) than other combination algorithms.

Figure 9 shows the overall comparison between the algorithms in terms of the optimization toolbox call counts. Similar to the previous results obtained with simple linear kernels, the two-step algorithms need to solve more optimization problems than the other combination algorithms.

5. 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 and therefore can be pruned, and 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 and each has its own (possibly multiple) kernel(s). In such a case, a good procedure for kernel combination implies 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. The dimensions we compare the existing MKL algorithms are the learning method, the functional form, the target function, the training method, the base learner, and the computational complexity. Then by looking at these dimensions, we form 12 groups of MKL variants to allow an organized discussion of the literature.

We also perform 10 experiments on four real data sets with simple linear kernels and eight experiments on three real data sets with complex Gaussian kernels comparing 16 MKL algorithms in practice. When combining simple linear kernels, in terms of accuracy, we see that using multiple kernels is better than using a single one but that in combination, trained linear combination is not always better than an untrained, unweighted combination and that nonlinear or data-dependent combination seem more promising. When combining complex Gaussian kernels, trained linear combination is better than nonlinear and data-dependent combinations but not than unweighted combination. Some MKL variants may be preferred because they use fewer support vectors or fewer kernels or need fewer calls to the optimizer during training. The relative importance of these criteria depend on the application at hand.

We conclude that multiple kernel learning is useful in practice and that there is ample evidence that better MKL algorithms can be devised for improved accuracy, decreased complexity and training time.

Acknowledgments

The authors would like to thank the editor and the three anonymous reviewers for their constructive comments, which significantly improved the presentation of the paper. This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, Boğaziçi University Scientific Research Project 07HA101 and the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant EEEAG

107E222. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK. M. Gönen is currently at the Department of Information and Computer Science, Aalto University School of Science and the Helsinki Institute for Information Technology (HIIT), Finland.

Appendix A. List of Acronyms

GMKL	Generalized Multiple Kernel Learning
GP	Gaussian Process
KFDA	Kernel Fisher Discriminant Analysis
KL	Kullback-Leibler
KRR	Kernel Ridge Regression
LMKL	Localized Multiple Kernel Learning
LP	Linear Programming
MKL	Multiple Kernel Learning
QCQP	Quadratically Constrained Quadratic Programming
QP	Quadratic Programming
RKDA	Regularized Kernel Discriminant Analysis
SDP	Semidefinite Programming
SILP	Semi-infinite Linear Programming
SMKL	Sparse Multiple Kernel Learning
SOCP	Second-order Cone Programming
SVM	Support Vector Machine
SVR	Support Vector Regression

Appendix B. List of 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
$\mathbb{S}^N$	Real symmetric $N \times N$ matrices
$\mathbb{N}$	Natural numbers
$\mathbb{Z}$	Integers
$\mathbb{Z}_+$	Nonnegative integers
$\|\mathbf{x}\|_p$	The ℓ_p-norm of vector $\mathbf{x}$
$\langle \mathbf{x}, \mathbf{y} \rangle$	Dot product between vectors $\mathbf{x}$ and $\mathbf{y}$
$k(\mathbf{x}, \mathbf{y})$	Kernel function between $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{K}$	Kernel matrix
$\mathbf{X}^\top$	Transpose of matrix $\mathbf{X}$
$\operatorname{tr}(\mathbf{X})$	Trace of matrix $\mathbf{X}$
$\|\mathbf{X}\|_F$	Frobenius norm of matrix $\mathbf{X}$
$\mathbf{X} \odot \mathbf{Y}$	Element-wise product between matrices $\mathbf{X}$ and $\mathbf{Y}$

References

Ethem Alpaydın. Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, 1999.

Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceeding 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. Consistency of the group Lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

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 and Alain Rakotomamonjy. Second order optimization of kernel parameters. In *NIPS Workshop on Automatic Selection of Optimal Kernels*, 2008.

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

Mario Christoudias, Raquel Urtasun, and Trevor Darrell. Bayesian localized multiple kernel learning. Technical Report UCB/EECS-2009-96, University of California at Berkeley, 2009.

Domenico Conforti and Rosita Guido. Kernel based support vector machine via semidefinite programming: Application to medical diagnosis. *Computers and Operations Research*, 37(8):1389–1394, 2010.

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 Rostamizadeh Afshin. Two-stage learning kernel algorithms. In *Proceedings of the 27th International Conference on Machine Learning*, 2010a.

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

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.

Isaac Martín de Diego, Alberto Muñoz, and Javier M. Moguerza. Methods for the combination of kernel matrices within a support vector framework. *Machine Learning*, 78(1–2):137–174, 2010a.

Isaac Martín de Diego, Ángel Serrano, Cristina Conde, and Enrique Cabello. Face verification with a kernel fusion method. *Pattern Recognition Letters*, 31:837–844, 2010b.

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.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

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.

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*, 2010a.

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Non-sparse regularization and efficient training with multiple kernels. Technical report, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2010b.

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, 2004a.

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, 2004b.

Gert R. G. Lanckriet, Minghua Deng, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. Kernel-based data fusion and its application to protein function prediction in Yeast. In *Proceedings of the Pacific Symposium on Biocomputing*, 2004c.

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

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

Chris Longworth and Mark 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 106)*. MOSEK ApS, Denmark, 2011.

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

William Stafford Noble. Support vector machine applications in computational biology. In Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors, *Kernel Methods in Computational Biology*, chapter 3. The MIT Press, 2004.

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.

Süreyya Özöğür-Akyüz and Gerhard Wilhelm Weber. Learning with infinitely many kernels via semi-infinite programming. In *Proceedings of Euro Mini Conference on Continuous Optimization and Knowledge-Based Technologies*, 2008.

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.

Jagarlapudi Saketha Nath, Govindaraj Dinesh, Sankaran Raman, Chiranjib Bhattacharya, Aharon Ben-Tal, and Kalpathi R. Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. In *Advances in Neural Information Processing Systems 22*, 2010.

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*, 32(5):788–798, 2010.

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

Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. *Machine Learning*, 79(1–2):73–103, 2010.

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. More generality in efficient multiple kernel learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

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

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems 13*, 2001.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

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.

Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group Lasso. In *Proceedings of the 27th International Conference on Machine Learning*, 2010a.

Zenglin Xu, Rong Jin, Shenghuo Zhu, Michael R. Lyu, and Irwin King. Smooth optimization for effective multiple kernel learning. In *Proceedings of the 24th AAAI Conference on Artifical Intelligence*, 2010b.

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*, 2009.

Jingjing Yang, Yuanning Li, Yonghong Tian, Ling-Yu Duan, and Wen Gao. Group-sensitive multiple kernel learning for object categorization. In *Proceedings of the 12th IEEE International Conference on Computer Vision*, 2009a.

Jingjing Yang, Yuanning Li, Yonghong Tian, Ling-Yu Duan, and Wen Gao. A new multiple kernel approach for visual concept learning. In *Proceedings of the 15th International Multimedia Modeling Conference*, 2009b.

Jingjing Yang, Yuanning Li, Yonghong Tian, Ling-Yu Duan, and Wen Gao. Per-sample multiple kernel approach for visual concept learning. *EURASIP Journal on Image and Video Processing*, 2010.

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.