More Generality in Efficient Multiple Kernel Learning

Manik Varma MANIK@MICROSOFT.COM
Microsoft Research India, Second Main Road, Sadashiv Nagar, Bangalore 560 080, India

Bodla Rakesh Babu RAKESHBABU@RESEARCH.IIIT.NET
CVIT, International Institute of Information Technology Hyderabad, Gachibowli, Hyderabad 500 032, India

Abstract

Recent advances in Multiple Kernel Learning (MKL) have positioned it as an attractive tool for tackling many supervised learning tasks. The development of efficient gradient descent based optimization schemes has made it possible to tackle large scale problems. Simultaneously, MKL based algorithms have achieved very good results on challenging real world applications. Yet, despite their successes, MKL approaches are limited in that they focus on learning a linear combination of given base kernels.

In this paper, we observe that existing MKL formulations can be extended to learn general kernel combinations subject to general regularization. This can be achieved while retaining all the efficiency of existing large scale optimization algorithms. To highlight the advantages of generalized kernel learning, we tackle feature selection problems on benchmark vision and UCI databases. It is demonstrated that the proposed formulation can lead to better results not only as compared to traditional MKL but also as compared to state-of-the-art wrapper and filter methods for feature selection.

1. Introduction

Support Vector Machines are basic tools in machine learning that are used for tasks such as classification, regression, *etc.* They find application in diverse areas ranging from vision to bioinformatics to natural-language processing. The success of SVMs in these fields is often dependent on the choice of a good kernel and features – ones that are typically hand-crafted and fixed in advance. However, hand-tuning kernel parameters can be difficult as can selecting and combining appropriate sets of features.

Multiple Kernel Learning (MKL) seeks to address this issue by learning the kernel from training data. In particular, it focuses on how the kernel can be learnt as a linear combination of given base kernels. Many MKL formulations have been proposed in the literature. In (Sonnenburg et al., 2006), it was shown that the MKL Block l_1 formulation of (Bach et al., 2004) could be expressed as a Semi-infinite Linear Program. Column generation methods and existing SVM solvers could then be used for efficient optimization and to tackle large scale problems involving as many as a million data points. Gradient descent can be more efficient than solving a series of linear programs and (Rakotomamonjy et al., 2008) demonstrated that training time could be further reduced by nearly an order of magnitude on some standard UCI datasets when the number of kernels was large. Simultaneously, MKL based algorithms have achieved very good results on bioinformatics and computer vision applications. These methods established the viability of MKL as a tool for tackling challenging real world problems.

Nevertheless, MKL approaches are limited in that they focus on learning linear combinations of base kernels – corresponding to the concatenation of individual kernel feature spaces. Far richer representations can be achieved by combining kernels in other fashions. For example, taking products of kernels corresponds to taking a tensor product of their feature spaces. This leads to a much higher dimensional feature representation as compared to feature concatenation. Furthermore, by focusing mainly on feature concatenation, MKL approaches do not consider the fundamental question of what are appropriate feature representations for a given task.

Some attempts have been made at addressing this issue. Most recently, (Bach, 2008) develop an innovative way of learning linear combinations of an exponential number of kernels of a certain type. The method can

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

therefore be leveraged in situations where the desired kernel combination can be approximated as a linear combination of appropriately chosen base kernels.

In this paper, we observe that it is fairly straight forward to extend traditional MKL formulations to handle generic kernel combinations. Furthermore, the gradient descent optimization developed and used in (Bach, 2008; Chapelle et al., 2002; Rakotomamonjy et al., 2008; Varma & Ray, 2007) can still be applied out of the box. It is therefore possible to learn rich feature representations without having to sacrifice any of the advantages of a well developed, large scale optimization toolkit. In addition to the kernel function, it is also possible to generalize the regularization on the kernel parameters. This can be used to incorporate prior knowledge about the kernel parameters if available. However, the price that one has to pay for such generality, is that the new MKL formulation is no longer convex. Nevertheless, we feel that the ability to explore appropriate feature representation is probably more important than being able to converge to the global optimum (of an inappropriate representation). This is borne out by our experimental results.

We highlight the advantages of generalized kernel learning by tackling feature selection problems on benchmark computer vision tasks, such as gender identification, as well as on some UCI datasets. We show that for a fixed number of selected features, standard MKL can lag behind our formulation by as much as 10% in some cases. Stated in another way, our formulation is capable of reaching the same classification accuracy as MKL with only a sixth of the features. We also present comparative results with AdaBoost, OWL-QN (Andrew & Gao, 2007), LP-SVM (Fung & Mangasarian, 2002), Sparse SVM (Chan et al., 2007) and BAHSIC (Song et al., 2007).

The rest of the paper is organized as follows: In Section 2 we review the development of MKL from initial work to current state-of-the-art. We then present our formulation in Section 3. The formulation is quite general and can be used for kernel combination, kernel parameter tuning, non-linear feature selection and dimensionality reduction. Since our empirical focus is on feature selection we review other feature selection methods in Section 4 and present comparative results to them in Section 5. We conclude in Section 6.

2. Related Work

Some of the earliest work on MKL was developed in (Crammer et al., 2002; Cristianini et al., 2001). Their focus was on optimizing loss functions such as kernel target alignment rather than the specific classification or regression problem at hand. This was addressed in the influential work of (Lanckriet et al., 2004) which showed how MKL could be formulated appropriately for a given task and optimized as an SDP or QCQP (for non-negative kernel weights). Nevertheless, QCQPs do not scale well to large problems and one of the first practical MKL algorithms was presented in (Bach et al., 2004). The block l_1 formulation, in conjunction with M-Y regularization, was developed so that efficient gradient descent could be performed using the SMO algorithm while still generating a sparse solution.

As already mentioned in the introduction, (Sonnenburg et al., 2006) retained the block l_1 regularization but reformulated the problem as a SILP. This made it applicable to large scale problems and the authors were impressively able to train their algorithm on a million splice data set. Further efficiency was obtained in (Rakotomamonjy et al., 2008; Varma & Ray, 2007) via gradient descent optimization and (Bach, 2008) opened up the possibility of training on an exponentially large number of kernels. Other interesting approaches have been proposed in (Argyriou et al., 2005; Ong et al., 2005; Zien & Ong, 2007) and include Hyperkernels and multi-class MKL.

Note that these methods essentially learn linear combinations of base kernels subject to l_1, or sometimes l_2 (Cristianini et al., 2001; Kloft et al., 2008), regularization of the kernel parameters. Most formulations are convex or can be made so by a change of variables.

3. Generalized MKL

Our objective is to learn a function of the form $f(\mathbf{x}) = \mathbf{w}^t\boldsymbol{\phi}_{\mathbf{d}}(\mathbf{x}) + b$ with the kernel $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^t_{\mathbf{d}}(\mathbf{x}_i)\boldsymbol{\phi}_{\mathbf{d}}(\mathbf{x}_j)$ representing the dot product in feature space $\boldsymbol{\phi}$ parameterized by $\mathbf{d}$. The function can be used directly for regression or the sign of the function can be used for classification. The goal in SVM learning is to learn the globally optimal values of $\mathbf{w}$ and b from training data $\{(\mathbf{x}_i, y_i)\}$. In addition, MKL also estimates the kernel parameters $\mathbf{d}$. We extend the MKL formulation of (Varma & Ray, 2007) to

$$\min_{\mathbf{w},b,\mathbf{d}} \quad \tfrac{1}{2}\mathbf{w}^t\mathbf{w} + \sum_i l(y_i, f(\mathbf{x}_i)) + r(\mathbf{d}) \quad (1)$$

$$\text{subject to} \quad \mathbf{d} \geq 0 \quad (2)$$

where both the regularizer r and the kernel can be any general differentiable functions of $\mathbf{d}$ with continuous derivative and l could be one of various loss functions such as $l = C\ \max(0, 1 - y_i f(\mathbf{x}_i))$ for classification or $l = C\ \max(0, |y_i - f(\mathbf{x}_i)| - \epsilon)$ for regression.

Three things are worth noting about the primal. First, we choose to use a non-convex formulation, as opposed to the convex $\sum_k \mathbf{w}_k^t \mathbf{w}_k / d_k$ (Rakotomamonjy et al., 2008), since for general kernel combinations, $\mathbf{w}_k^t \mathbf{w}_k$ need not tend to zero when d_k tends to zero. Second, we place $r(\mathbf{d})$ in the objective and incorporate a scale parameter within it rather than having it as an equality constraint (typically $\sum_k d_k = 1$ or $\sum_k d_k^2 = 1$). Third, the constraint $\mathbf{d} \geq 0$ can often be relaxed to a more general one which simply requires the learnt kernel to be positive definite. Conversely, the constraints can also be strengthened if prior knowledge is available. In either case, if $\nabla_{\mathbf{d}} r$ exists then the gradient descent based optimization is still applicable. However, the projection back into the feasible set can get more expensive.

In order to leverage existing large scale optimizers, we follow the standard procedure (Chapelle et al., 2002) of reformulating the primal as a nested two step optimization. In the outer loop, the kernel is learnt by optimizing over $\mathbf{d}$ while, in the inner loop, the kernel is held fixed and the SVM parameters are learnt. This can be achieved by rewriting the primal as follows

$$\begin{aligned} \min_{\mathbf{d}} \quad & T(\mathbf{d}) \quad \text{subject to} \quad \mathbf{d} \geq 0 && (3) \\ \textit{where} \quad & T(\mathbf{d}) = \min_{\mathbf{w},b} \tfrac{1}{2}\mathbf{w}^t\mathbf{w} + \sum_i l(y_i, f(\mathbf{x}_i)) + r(\mathbf{d}) \end{aligned}$$

We now need to prove that $\nabla_{\mathbf{d}} T$ exists, and calculate it efficiently, if we are to utilize gradient descent in the outer loop. This can be achieved by moving to the dual formulation of T given by (for classification and regression respectively)

$$\begin{aligned} W_C(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^t\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t\mathbf{Y}\mathbf{K}_{\mathbf{d}}\mathbf{Y}\boldsymbol{\alpha} + r(\mathbf{d}) && (4) \\ \text{subject to} \quad & \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} = 0, \quad 0 \leq \boldsymbol{\alpha} \leq C && (5) \end{aligned}$$

and

$$\begin{aligned} W_R(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t\mathbf{K}_{\mathbf{d}}\boldsymbol{\alpha} \\ & + r(\mathbf{d}) - \epsilon\mathbf{1}^t|\boldsymbol{\alpha}| && (6) \\ \text{subject to} \quad & 1^t\boldsymbol{\alpha} = 0, \quad 0 \leq |\boldsymbol{\alpha}| \leq C && (7) \end{aligned}$$

where $\mathbf{K}_{\mathbf{d}}$ is the kernel matrix for a given $\mathbf{d}$ and $\mathbf{Y}$ is a diagonal matrix with the labels on the diagonal.

Note that we can write $T = r + P$ and $W = r + D$ with strong duality holding between P and D. Therefore, $T(\mathbf{d}) = W(\mathbf{d})$ for any given value of $\mathbf{d}$, and it is sufficient for us to show that W is differentiable and calculate $\nabla_{\mathbf{d}} W$. Proof of the differentiability of W_C and W_R comes from Danskin's Theorem (Danskin, 1967). Since the feasible set is compact, the gradient can be shown to exist if k, r, $\nabla_{\mathbf{d}} k$ and $\nabla_{\mathbf{d}} r$ are smoothly varying functions of $\mathbf{d}$ and if $\boldsymbol{\alpha}^*$, the value of $\boldsymbol{\alpha}$ that optimizes W, is unique. Furthermore, a straight forward extension of Lemma 2 in (Chapelle et al., 2002) can be used to show that W_C and W_R (as well as others obtained from loss functions for novelty detection, ranking, *etc.*) have derivatives given by

$$\frac{\partial T}{\partial d_k} = \frac{\partial W}{\partial d_k} = \frac{\partial r}{\partial d_k} - \tfrac{1}{2}\boldsymbol{\alpha}^{*t}\frac{\partial \mathbf{H}}{\partial d_k}\boldsymbol{\alpha}^* \qquad (8)$$

where $\mathbf{H} = \mathbf{YKY}$ for classification and $\mathbf{H} = \mathbf{K}$ for regression. Thus, in order to take a gradient step, all we need to do is obtain $\boldsymbol{\alpha}^*$. Note that since W_C or W_R are equivalent to their corresponding single kernel SVM duals with kernel matrix $\mathbf{K}_{\mathbf{d}}$, $\boldsymbol{\alpha}^*$ can be obtained by any SVM optimization package. The final algorithm is given in Algorithm 1 and we refer to it as Generalized MKL (GMKL). The step size s^n is chosen based on the Armijo rule to guarantee convergence and the projection step, for the constraints $\mathbf{d} \geq 0$, is as simple as $\mathbf{d} \leftarrow \max(\mathbf{0}, \mathbf{d})$. Note that the algorithm is virtually unchanged from (Varma & Ray, 2007) apart from the more general form of the kernel k and regularizer r. If a faster rate of convergence was required, our assumptions could be suitably modified so as to take second order steps rather than perform gradient descent.

Only very mild restrictions have been placed on the form of the learnt kernel k and regularizer r. As regards k, the only constraints that have been imposed are that $\mathbf{K}$ be strictly positive definite for all valid $\mathbf{d}$ and that $\nabla_{\mathbf{d}} k$ exists and be continuous. Many kernels can be constructed that satisfy these properties. One can, of course, learn the standard sum of base kernels. More generally, products of base kernels, and other combinations which yield positive definite kernels, can also be learnt now. In addition, one can also tune kernel parameters in general kernels such as $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = (d_0 + \sum_m d_m \mathbf{x}_i^t \mathbf{A}_m \mathbf{x}_j)^n$ or

Algorithm 1 Generalized MKL.

1: $n \leftarrow 0$
2: Initialize $\mathbf{d}^0$ randomly.
3: **repeat**
4: $\quad \mathbf{K} \leftarrow k(\mathbf{d}^n)$
5: $\quad$ Use an SVM solver of choice to solve the single kernel problem with kernel $\mathbf{K}$ and obtain $\boldsymbol{\alpha}^*$.
6: $\quad d_k^{n+1} \leftarrow d_k^n - s^n \left(\frac{\partial r}{\partial d_k} - \frac{1}{2}\boldsymbol{\alpha}^{*t}\frac{\partial \mathbf{H}}{\partial d_k}\boldsymbol{\alpha}^* \right)$
7: $\quad$ Project $\mathbf{d}^{n+1}$ onto the feasible set if any constraints are violated.
8: $\quad n \leftarrow n + 1$
9: **until** converged

$k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\sum_m d_m \mathbf{x}_i^t \mathbf{A}_m \mathbf{x}_j}$. Combined with a sparsity promoting regularizer on $\mathbf{d}$, this can be used for non-linear dimensionality reduction and feature selection for appropriate choices of $\mathbf{A}$. Note, however, that such kernels do not lead to convex formulations.

As regards r, we only require that its derivative should exist and be continuous. Since $\mathbf{d}$ can be restricted to the non-negative orthant, various forms of p-norm regularisers with $p \geq 1$ fall in this category. In particular, l_1 regularization with $r(\mathbf{d}) = \boldsymbol{\sigma}^t \mathbf{d}$ or variations of (Chan et al., 2007) could be used for learning sparse solutions. Alternatively, l_2 regularization of the form $r(\mathbf{d}) = (\mathbf{d} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{d} - \boldsymbol{\mu})$ can be used when only a small number of relevant kernels are present or if prior knowledge in the form of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is available (for instance, from transfer learning). Finally, for regression, one could incorporate the term $\log |K_{\mathbf{d}}|$ into r so as to obtain a MAP estimate of a probabilistic formulation. The naïve substitution $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$ would then render our formulation near identical to a marginal likelihood approach in Gaussian Processes.

4. Feature Selection Methods

We compare our formulation to traditional MKL as well as the following feature selection methods

Boosting : The LPBoost formulation of (Bi et al., 2004) is similar to that of standard MKL and boosting generalizes standard MKL's decision function. Boosting can therefore be used to learn linear combinations of base kernels. Individual "weak classifier" SVMs are pre-learnt from each of the given base kernels and combined using AdaBoost. This can be attractive when there are a large number of kernels or when kernels are made available incrementally. While the computational costs are low, the empirical results were found to be poor as the learnt kernel weights could not influence the pre-learnt weak classifiers. Of course, in traditional boosting, the weak classifiers and the weights are learnt together and we present comparative results to (Baluja & Rowley, 2007) which represents a state-of-the-art boosting method for gender identification.

OWL-QN (Andrew & Gao, 2007) : This is a large scale implementation of l_1 logistic regression. The method learns a function of the form $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x}$ by minimizing $(1/C)||w||_1 + \sum_i l(y_i, f(\mathbf{x}_i))$ where l is the log loss. Despite being linear, the method can sometimes outperform boosting. Nevertheless, the overall performance is poor as compared to the other linear methods since OWL-QN does not have an explicit bias term. One could simulate a bias by adding a constant feature but the corresponding weight could be set to zero due to the l_1 regularization. When this doesn't happen, OWL-QN's performance is comparable to LP-SVM and Sparse-SVM.

LP-SVM (Fung & Mangasarian, 2002) : This is the standard SVM formulation but with the l_2 regularization on $\mathbf{w}$ replaced by l_1 regularization. We consider the linear formulation which learns a function of the form $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$ by minimizing $||w||_1 + C\sum_i l(y_i, f(\mathbf{x}_i))$ where l is the hinge loss. Seeing that the hinge loss is very similar to the log loss, the formulation appears to be very similar to OWL-QN. However, due to the explicit bias term b which is not included in the l_1 regularization, LP-SVM can sometimes perform much better than OWL-QN. Somewhat surprisingly, the performance of the linear LP-SVM could even be better than that of non-linear MKL (though not GMKL).

Sparse-SVM (Chan et al., 2007) : This method does not use explicit l_1 regularization to enforce sparsity. Instead, it places a direct cardinality constraint on the hyperplane normal. It learns a function of the form $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$ by minimizing $||w||_2 + C\sum_i l(y_i, f(\mathbf{x}_i))$ subject to $||\mathbf{w}||_0 \leq r$, where l is the hinge loss. We take the convex QCQP relaxation (QCQP-SSVM) proposed by the authors. Empirically, we found the performance of Sparse-SVM to be very similar to that of LP-SVM though, being a QCQP, it took much longer to optimize.

BAHSIC (Song et al., 2007) : This is a leading filter method which runs a backward selection algorithm discarding features based on their label dependence as measured by the Hilbert-Schmidt independence criterion. We use an RBF kernel for the data (the same as used by boosting, MKL and GMKL) and a linear kernel for the labels. BAHSIC outputs a ranked list of features from which a subset of the desired size can be selected. An SVM with an RBF kernel is then trained using the selected features. In our experiments, we found backward selection to be computationally very expensive without offering any advantages in terms of classification accuracy. Given identical kernels, BAHSIC performed substantially worse than GMKL,

5. Experiments

In this section we evaluate generalized kernel learning on various feature selection problems. We investigate gender identification from frontal facial images using as few pixels as possible. It is demonstrated that, on the benchmark database of (Moghaddam & Yang, 2002), GMKL can outperform MKL by as much as 10% for a given number of features. Similarly, on the

Table 1: Gender identification results. The final row summarizes the average number of features selected (in brackets) by each wrapper method and the resultant classification accuracy. See text for details.

N_d	AdaBoost	B&R 2007	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
10	76.3 ± 0.9	79.5 ± 1.9	71.6 ± 1.4	84.9 ± 1.9	79.5 ± 2.6	81.2 ± 3.2	80.8 ± 0.2	$\mathbf{88.7 \pm 0.8}$
20	–	82.6 ± 0.6	80.5 ± 3.3	87.6 ± 0.5	85.6 ± 0.7	86.5 ± 1.3	83.8 ± 0.7	$\mathbf{93.2 \pm 0.9}$
30	–	83.4 ± 0.3	84.8 ± 0.4	89.3 ± 1.1	88.6 ± 0.2	89.4 ± 2.4	86.3 ± 1.6	$\mathbf{95.1 \pm 0.5}$
50	–	86.9 ± 1.0	88.8 ± 0.4	90.6 ± 0.6	89.5 ± 0.2	91.0 ± 1.3	89.4 ± 0.9	$\mathbf{95.5 \pm 0.7}$
80	–	88.9 ± 0.6	90.4 ± 0.2	–	90.6 ± 1.1	92.4 ± 1.4	90.5 ± 0.2	–
100	–	89.5 ± 0.2	90.6 ± 0.3	–	90.5 ± 0.2	94.1 ± 1.3	91.3 ± 1.3	–
150	–	91.3 ± 0.5	90.3 ± 0.8	–	90.7 ± 0.2	94.5 ± 0.7	–	–
252	–	93.1 ± 0.5	–	–	90.8 ± 0.0	94.3 ± 0.1	–	–
	76.3 (12.6)	–	91 (221.3)	91 (58.3)	90.8 (252)	–	91.6 (146.3)	95.5 (69.6)

UCI datasets, there can be as much as a 6% to 10% difference in performance between GMKL and MKL. We also demonstrate that GMKL performs better than the other methods considered.

To generate feature selection results, we can vary the hyper-parameter C in the wrapper methods to select a desired number of features. However, this strategy does not yield good classification results even though globally optimal solutions are obtained. Low values of C encouraged greater sparsity but also permitted more classification errors. We obtained much better results by the theoretically suboptimal strategy of fixing C to a large value (chosen via cross-validation so as to minimize classification error), learning the classifier, taking the top ranked components of $\mathbf{w}$ (or $\mathbf{d}$) and relearning the classifier using only the selected features.

This technique was used to generate the results in Tables 1 and 2. For each dataset, the very last row summarizes the number of features selected (N_s) by each wrapper method and the resultant classification accuracy. When the number of desired features (N_d) is less than N_s, the classification accuracy is determined using the N_d top ranked features. Otherwise, when $N_d > N_s$, the table entry is left blank as the classification accuracy either plateaus or decreases as suboptimal features are added. In such a situation, it is better to choose only N_s features and maintain accuracy.

5.1. Gender Identification

We tackle the binary gender identification problem on the benchmark database of (Moghaddam & Yang, 2002). The database has images of 1044 male and 711 female faces giving a total of 1755 images in all. We follow the standard experimental setup and use 1053 images for training and 702 for testing. Results are averaged over 3 random splits of the data.

Each image in the database has been pre-processed by (Moghaddam & Yang, 2002) to be aligned and has been scaled down to have dimensions 21×12. Thus, each image has 252 pixels and we associate an RBF kernel with each pixel based on its grey scale intensity directly. For Generalized MKL, the 252 base kernels are combined by taking their product to get $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{m=1}^{252} e^{-d_m(x_{im}-x_{jm})^2}$ where x_{im} and x_{jm} represent the intensity of the m^{th} pixel in image i and j respectively. For standard MKL, the same 252 base kernels are combined using the sum representation to get $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{252} d_m e^{-\gamma_m(x_{im}-x_{jm})^2}$. Both methods are subject to l_1 regularization on $\mathbf{d}$ so that only a few kernels are selected. AdaBoost can also be used to combine the 252 weak classifiers derived from the individual base kernels. The method of (Baluja & Rowley, 2007) can be considered as a state-of-the-art boosting variant for this problem and operates on pairs of pixels. We use RBF kernels for BAHSIC as well while the other methods are linear.

Table 1 lists the feature selection results. AdaBoost tended to perform the worst and selected only 12.6 features on average. The poor performance was due to the fact that each of the 252 SVMs was learnt independently. The weak classifier coefficients (i.e. kernel weights) did not influence the individual SVM parameters. By contrast, there is a very tight coupling between the two in MKL and GMKL and this ensures better performance. Of course, other forms of boosting do not have this limitation and the state-of-the-art boosting method of (Baluja & Rowley, 2007) performs better but is still significantly inferior to GMKL.

The performance of the linear feature selection methods is quite variable. Logistic regression performs poorly as the implicit bias weight got set to zero due to the l_1 regularization. On the other hand, LP-SVM and Sparse-SVM perform even better than standard MKL (though not better than GMKL). BAHSIC, which is a non-linear filter method, follows the same

Table 2: UCI results with datasets having N training points and M features. See text for details.

Ionosphere: $N = 246$, $M = 34$, Uniform MKL $= 89.9 \pm 2.5$, Uniform GMKL $= 94.6 \pm 2.0$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
5	75.2 ± 6.9	84.0 ± 6.0	86.7 ± 3.1	87.0 ± 3.1	87.1 ± 3.6	85.1 ± 3.2	$\mathbf{90.9 \pm 1.9}$
10	–	87.6 ± 2.2	90.6 ± 3.4	90.2 ± 3.5	90.2 ± 2.6	87.8 ± 2.4	$\mathbf{93.7 \pm 2.1}$
15	–	89.1 ± 1.9	93.0 ± 2.1	91.9 ± 2.0	92.6 ± 3.0	87.7 ± 2.2	$\mathbf{94.1 \pm 2.1}$
20	–	89.2 ± 1.8	92.8 ± 3.0	92.4 ± 2.5	93.4 ± 2.6	87.8 ± 2.8	–
25	–	89.1 ± 1.9	92.6 ± 2.7	92.4 ± 2.7	94.0 ± 2.2	87.9 ± 2.7	–
30	–	–	92.6 ± 2.6	92.9 ± 2.5	94.3 ± 1.9	–	–
34	–	–	92.6 ± 2.6	92.9 ± 2.5	$\mathbf{94.6 \pm 2.0}$	–	–
	75.1 (9.8)	89.2 (25.2)	92.6 (34.0)	92.9 (34.0)	–	88.1 (29.3)	94.4 (16.9)

Parkinsons: $N = 136$, $M = 22$, Uniform MKL $= 87.3 \pm 3.9$, Uniform GMKL $= 91.0 \pm 3.5$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
3	79.4 ± 6.5	81.7 ± 2.7	76.4 ± 4.5	76.1 ± 5.8	85.2 ± 3.8	83.7 ± 4.4	$\mathbf{86.3 \pm 4.1}$
7	–	82.6 ± 3.3	86.2 ± 2.7	86.1 ± 4.0	88.5 ± 3.6	84.7 ± 5.2	$\mathbf{92.6 \pm 2.9}$
11	–	83.5 ± 2.8	86.0 ± 3.5	86.1 ± 3.1	89.4 ± 3.6	86.3 ± 4.3	–
15	–	–	87.0 ± 3.3	86.3 ± 3.1	89.9 ± 3.5	–	–
22	–	–	87.2 ± 3.2	87.2 ± 3.0	91.0 ± 3.5	–	–
	80.2 (5.2)	83.6 (11.1)	87.2 (22.0)	87.2 (22.0)	–	88.3 (14.6)	92.7 (9.0)

Musk: $N = 333$, $M = 166$, Uniform MKL $= 90.2 \pm 3.2$, Uniform GMKL $= 93.8 \pm 1.9$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
10	64.2 ± 4.0	72.8 ± 2.9	69.8 ± 5.1	72.6 ± 3.7	76.5 ± 3.5	80.0 ± 3.0	$\mathbf{81.1 \pm 3.8}$
20	65.5 ± 4.1	76.0 ± 4.4	73.8 ± 4.9	76.7 ± 4.1	83.6 ± 3.3	84.5 ± 3.4	$\mathbf{89.9 \pm 2.3}$
30	65.4 ± 4.1	80.8 ± 2.5	79.0 ± 2.8	79.4 ± 3.0	86.7 ± 2.8	86.2 ± 3.3	$\mathbf{92.6 \pm 1.7}$
40	–	81.6 ± 2.9	81.5 ± 3.2	81.8 ± 2.8	87.4 ± 2.8	87.0 ± 3.2	$\mathbf{93.3 \pm 2.0}$
60	–	83.0 ± 1.9	83.6 ± 2.8	83.5 ± 2.4	90.0 ± 2.6	87.8 ± 3.3	–
100	–	–	83.4 ± 2.9	83.3 ± 2.5	$\mathbf{93.6 \pm 1.8}$	–	–
166	–	–	83.4 ± 2.9	83.3 ± 2.5	$\mathbf{93.8 \pm 1.9}$	–	–
	65.5 (31.1)	83.5 (86.7)	83.4 (166.0)	83.3 (166.0)	–	88.2 (73.2)	93.6 (57.9)

Sonar: $N = 145$, $M = 60$, Uniform MKL $= 82.9 \pm 3.4$, Uniform GMKL $= 84.6 \pm 4.1$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
5	64.6 ± 6.6	68.9 ± 5.6	68.0 ± 7.9	68.4 ± 6.2	61.1 ± 6.6	70.4 ± 4.5	$\mathbf{74.4 \pm 5.1}$
10	67.9 ± 6.4	68.7 ± 4.6	71.5 ± 5.4	70.9 ± 5.9	73.1 ± 6.1	74.6 ± 5.6	$\mathbf{80.2 \pm 4.9}$
15	67.3 ± 6.4	71.4 ± 3.6	71.4 ± 3.3	72.2 ± 4.5	74.7 ± 7.7	76.5 ± 7.0	$\mathbf{80.7 \pm 5.5}$
20	–	73.1 ± 2.6	73.7 ± 2.8	74.0 ± 3.1	77.9 ± 5.7	79.5 ± 4.6	$\mathbf{82.0 \pm 5.3}$
25	–	73.5 ± 2.8	74.1 ± 3.5	73.6 ± 3.6	78.6 ± 5.2	81.1 ± 4.2	–
30	–	73.9 ± 3.2	73.4 ± 3.4	73.8 ± 3.9	80.8 ± 4.7	81.4 ± 4.2	–
40	–	–	73.6 ± 3.8	73.7 ± 3.8	81.4 ± 3.9	–	–
60	–	–	73.6 ± 3.6	73.5 ± 4.0	$\mathbf{84.6 \pm 4.1}$	–	–
	67.38 (18.7)	74.7 (39.3)	73.6 (60.0)	73.5 (60.0)	–	81.4 (38.6)	82.3 (20.4)

Wpbc: $N = 135$, $M = 34$, Uniform MKL $= 72.1 \pm 5.4$, Uniform GMKL $= 77.0 \pm 6.4$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
5	$\mathbf{76.7 \pm 2.2}$	74.2 ± 4.1	75.4 ± 2.7	75.5 ± 2.7	76.6 ± 2.1	67.8 ± 6.0	76.1 ± 3.8
10	–	77.2 ± 5.2	75.9 ± 4.6	76.5 ± 3.8	77.3 ± 2.3	68.7 ± 3.3	$\mathbf{77.8 \pm 3.3}$
15	–	77.8 ± 5.5	76.2 ± 5.1	77.2 ± 5.0	76.2 ± 0.0	69.4 ± 5.1	$\mathbf{78.3 \pm 3.6}$
20	–	78.1 ± 5.3	78.2 ± 5.2	77.7 ± 5.2	77.3 ± 6.3	70.1 ± 5.1	–
25	–	–	$\mathbf{79.1 \pm 6.2}$	77.8 ± 5.8	77.4 ± 6.4	–	–
34	–	–	79.0 ± 6.2	78.9 ± 5.6	77.0 ± 6.4	–	–
	76.7 (5.1)	78.3 (20.8)	79.0 (34.0)	78.9 (34.0)	–	69.3 (24.8)	80.0 (16.8)

trend and its performance is significantly worse than GMKL with identical kernels. This would suggest that wrapper methods based on the right feature representation should be preferable to filter methods which do not directly optimize for classification accuracy.

The comparison between MKL and GMKL is even starker. GMKL achieves a classification accuracy of 93.2% using as few as 20 features and 95.1% using only 30 features. In comparison, standard MKL achieves just 83.8% and 86.3% respectively. This reinforces the observation that choosing the right kernel representation is much more important than converging to the globally optimal solution. Finally, the MKL and GMKL results for fixed uniform weights chosen via cross-validation are 92.6 ± 0.9 and 94.3 ± 0.1 respectively. Note that these results are obtained using all 252 features. GMKL can obtain a similar classification accuracy using as few as 25 features.

In summary, there can be a difference of almost 10% between standard MKL and GMKL for a fixed small number of features. GMKL also does significantly better than BAHSIC and achieves nearly a 10x compression factor as compared to taking uniform weights.

5.2. UCI Datasets

We also present comparative results on UCI datasets. We follow the standard experimental methodology (Rakotomamonjy et al., 2008) where 70% of the points are used for training and the remaining 30% for testing. Results are reported over 20 random splits of the data. All datasets are preprocessed to have zero mean and unit variance.

An RBF kernel is assigned to each of the M features in a given dataset. The M RBF kernels are then combined linearly for standard MKL and by taking their product for GMKL. The learnt kernels are of the form $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{M} d_m e^{-\gamma_m (x_{im} - x_{jm})^2}$ and $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{m=1}^{M} e^{-d_m (x_{im} - x_{jm})^2}$ respectively.

Table 2 lists the feature selection results. The analysis is similar to that of gender identification except that OWL-QN performs better as the implicit bias is not always being set to zero. For a fixed number of features, GMKL has the best classification results even as compared to MKL or BAHSIC (though the variance can be high for all the methods). Furthermore, a kernel with fixed uniform weights yields ballpark classification accuracies though GMKL can achieve the same results using far fewer features.

5.3. Comparison to SimpleMKL and HMKL

The classification performance of standard MKL can be improved by adding extra base kernels which are either more informative or help better approximate a desired kernel function. However, this can lead to a more complex and costlier learning task. We therefore leave aside feature selection for the moment and compare our results to those reported in (Rakotomamonjy et al., 2008). Table 3 lists classification performance on the 5 datasets of (Rakotomamonjy et al., 2008). The number of kernels input to each method are reported in brackets. As can be seen, GMKL can achieve slightly better performance than SimpleMKL while training on far fewer kernels. Of course, one can reduce the number of kernels input to SimpleMKL but this will result in reduced accuracy. In the limit that only a single kernel is used per feature, we will get back the results of Table 2 where GMKL does much better than standard MKL.

The results for Liver were obtained using l_2 regularization. This lead to a 7% improvement in performance over SimpleMKL as a sparse representation is not suitable for this database (there are only 6 features). Pima also has very few features but l_1 and l_2 regularization give comparable results.

Finally, we also compare results to Hierarchical MKL (Bach, 2008). We use a quadratic kernel of the form $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \sum_m d_m x_{im} x_{jm})^2$ as compared to the more powerful $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_m (1 + x_{im} x_{jm})^4$ of (Bach, 2008) which decomposes to a linear combination of 5^M kernels. Nevertheless, as shown in Table 4, we achieve slightly better results with less computational cost.

Table 3: Comparison with the results in (Rakotomamonjy et al., 2008). GMKL achieves slightly better results but takes far fewer kernels as input.

Database	SimpleMKL	GMKL
Sonar	80.6 ± 5.1 (793)	$\mathbf{82.3 \pm 4.8}$ (**60**)
Wpbc	76.7 ± 1.2 (442)	$\mathbf{79.0 \pm 3.5}$ (**34**)
Ionosphere	91.5 ± 2.5 (442)	$\mathbf{93.0 \pm 2.1}$ (**34**)
Liver	65.9 ± 2.3 (091)	$\mathbf{72.7 \pm 4.0}$ (**06**)
Pima	76.5 ± 2.6 (117)	$\mathbf{77.2 \pm 2.1}$ (**08**)

Table 4: Comparison between HKL and GMKL.

Database	N	M	HKL	GMKL
Magic04	1024	10	84.4 ± 0.8	$\mathbf{86.2 \pm 1.2}$
Spambase	1024	57	91.9 ± 0.7	$\mathbf{93.2 \pm 0.8}$
Mushroom	1024	22	99.9 ± 0.2	$\mathbf{100 \pm 0.0}$

6. Conclusions

In this paper, we have shown how traditional MKL formulations can be easily extended to learn general kernel combinations subject to general regularization on the kernel parameters. While our focus was on binary classification our approach can be applied to other loss functions and even other formulations such as Local MKL, multi-class and multi-label MKL. Generalized kernel learning can be achieved very efficiently based on gradient descent optimization and existing large scale SVM solvers. As such, it is now possible to learn much richer feature representations as compared to standard MKL without sacrificing any efficiency in terms of speed of optimization.

Our GMKL formulation based on products of kernels was shown to give good results for various feature selection problems – not only as compared to traditional MKL but also as compared to leading wrapper and filter feature selection methods. Of course, taking products of kernels might not always be the right approach to every problem. In such cases, our formulation can be used to learn other appropriate representation including sums of kernels. Finally, it should be noted that the classification accuracy of GMKL with learnt weights tends to be much the same as that obtained using uniform weights chosen through cross-validation. The advantage in learning would therefore seem to lie in the fact that GMKL can learn to achieve the same classification accuracy but using far fewer features.

Acknowledgements

We are grateful to P. Anandan, Andrew Zisserman and our reviewers for providing helpful feedback.

References

Andrew, G., & Gao, J. (2007). Scalable training of L_1-regularized log-linear models. *ICML* (pp. 33–40).

Argyriou, A., Micchelli, C. A., & Pontil, M. (2005). Learning convex combinations of continuously parameterized basic kernels. *COLT* (pp. 338–352).

Bach, F. R. (2008). Exploring large feature spaces with hierarchical multiple kernel learning. *NIPS* (pp. 105–112).

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. *ICML* (pp. 6–13).

Baluja, S., & Rowley, H. (2007). Boosting sex identification performance. *IJCV*, *71*, 111–119.

Bi, J., Zhang, T., & Bennet, K. P. (2004). Column-generation boosting methods for mixture of kernels. *Proc. SIGKDD* (pp. 521–526).

Chan, A. B., Vasconcelos, N., & Lanckriet, G. (2007). Direct convex relaxations of sparse SVM. *ICML* (pp. 145–153).

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for Support Vector Machines. *Machine Learning*, *46*, 131–159.

Crammer, K., Keshet, J., & Singer, Y. (2002). Kernel design using boosting. *NIPS* (pp. 537–544).

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. (2001). On kernel-target alignment. *NIPS* (pp. 367–373).

Danskin, J. M. (1967). *The theorey of max-min and its applications to weapons allocation problems.*

Fung, G., & Mangasarian, O. L. (2002). *A feature selection newton method for support vector machine classification* (Technical Report 02-03). Univ. of Wisconsin.

Kloft, M., Brefeld, U., Laskov, P., & Sonnenburg, S. (2008). Non-sparse Multiple Kernel Learning. *NIPS Workshop on Kernel Learning.*

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *JMLR*, *5*, 27–72.

Moghaddam, B., & Yang, M. H. (2002). Learning gender with support faces. *IEEE PAMI*, *24*, 707–711.

Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *JMLR*, *6*, 1043–1071.

Rakotomamonjy, A., Bach, F., Grandvalet, Y., & Canu, S. (2008). Simplemkl. *JMLR*, *9*, 2491–2521.

Song, L., Smola, A., Gretton, A., Borgwardt, K., & Bedo, J. (2007). Supervised feature selection via dependence estimation. *ICML* (pp. 823–830).

Sonnenburg, S., Raetsch, G., Schaefer, C., & Schoelkopf, B. (2006). Large scale multiple kernel learning. *JMLR*, *7*, 1531–1565.

Varma, M., & Ray, D. (2007). Learning the discriminative power-invariance trade-off. *ICCV*.

Zien, A., & Ong, C. S. (2007). Multiclass multiple kernel learning. *ICML* (pp. 1191–1198).