SPG-GMKL: Generalized Multiple Kernel Learning with a Million Kernels

Ashesh Jain
IIT Delhi
asheshjain399@gmail.com

S. V. N. Vishwanathan
Purdue University
vishy@stats.purdue.edu

Manik Varma
Microsoft Research India
manik@microsoft.com

ABSTRACT

Multiple Kernel Learning (MKL) aims to learn the kernel in an SVM from training data. Many MKL formulations have been proposed and some have proved effective in certain applications. Nevertheless, as MKL is a nascent field, many more formulations need to be developed to generalize across domains and meet the challenges of real world applications. However, each MKL formulation typically necessitates the development of a specialized optimization algorithm. The lack of an efficient, general purpose optimizer capable of handling a wide range of formulations presents a significant challenge to those looking to take MKL out of the lab and into the real world.

This problem was somewhat alleviated by the development of the Generalized Multiple Kernel Learning (GMKL) formulation which admits fairly general kernel parameterizations and regularizers subject to mild constraints. However, the projected gradient descent GMKL optimizer is inefficient as the computation of the step size and a reasonably accurate objective function value or gradient direction are all expensive. We overcome these limitations by developing a Spectral Projected Gradient (SPG) descent optimizer which: a) takes into account second order information in selecting step sizes; b) employs a non-monotone step size selection criterion requiring fewer function evaluations; c) is robust to gradient noise, and d) can take quick steps when far away from the optimum.

We show that our proposed SPG-GMKL optimizer can be an order of magnitude faster than projected gradient descent on even small and medium sized datasets. In some cases, SPG-GMKL can even outperform state-of-the-art specialized optimization algorithms developed for a single MKL formulation. Furthermore, we demonstrate that SPG-GMKL can scale well beyond gradient descent to large problems involving a million kernels or half a million data points. Our code and implementation are available publically.

KDD'12, August 12–16, 2012, Beijing, China.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentations

Keywords

Multiple Kernel Learning, Support Vector Machines, Spectral Projected Gradient Descent

1. INTRODUCTION

Support Vector Machines (SVMs) have become ubiquitous in diverse areas ranging from computer vision to natural language processing to bioinformatics. One of the reasons for their widespread adoption is the availability of general purpose, efficient optimizers capable of handling various SVM formulations as demanded by real world applications.

The success of non-linear SVMs also depends on the choice of a good kernel. Multiple Kernel Learning (MKL) techniques aim to learn such a kernel from training data often as a combination of given base kernels. MKL can be used for various tasks such as learning a similarity measure tailored to the given application, heterogeneous feature combination and learning models with sparse structures. For instance, MKL techniques have yielded state-of-the-art results on very challenging object recognition [22] and object detection [30] problems in computer vision. They have also yielded superior results as compared to many state-of-the-art filter and wrapper methods for feature selection [28]. However, MKL techniques have not proved to be as popular as the baseline SVMs upon which they hope to improve.

The primary reason for this is that MKL is still a nascent field and, while a few formulations have shown promising results in certain contexts, many more need to be developed that are robust to over fitting and generalize across domains. At the same time, existing formulations need to be tried out on many real world applications in order to determine regimes in which they perform well. Unfortunately, there does not exist an efficient, general purpose optimizer which gives practitioners the flexibility to quickly prototype novel MKL formulations or try out existing ones on large scale applications. As things stand, the practitioner faces the significant challenge of having to come up with a different optimization technique for every novel MKL formulation that she might wish to explore.

This problem was somewhat alleviated by the development of the Generalized Multiple Kernel Learning (GMKL)

formulation [28]. The formulation allows learning fairly general kernel parameterizations, including linear and non-linear kernel combinations, subject to general regularization. The only restriction that is placed on the kernel and regularizer is that their derivative with respect to the kernel parameters must exist and be jointly continuous with the SVM dual variables $\boldsymbol{\alpha}$. Most of the kernel learning formulations proposed in the literature satisfy these constraints.

However, the general purpose, projected gradient descent based GMKL optimizer is not efficient and doesn't scale well. In particular, it has two significant shortcomings. First, the gradient descent step size is calculated via the Armijo rule to guarantee convergence. This necessitates the solving of many expensive, single kernel SVMs on the entire data set in order to take a single step. Second, the objective function value and the gradient descent direction are also obtained by solving an SVM. To avoid noise in these estimates the SVM Quadratic program (QP) has to be solved to a very high precision. Unfortunately, high precision solvers don't scale beyond toy problems, while large scale specialized SVM QP solvers based on chunking and decomposition [6] typically produce low precision results. Optimizing GMKL via projected gradient descent can thus be significantly slower than solving a single SVM.

Our objective, in this paper, is to speed up GMKL optimization by an order of magnitude in many cases. We achieve this by carefully designing an alternative optimizer based on Spectral Projected Gradient (SPG) descent [5]. SPG is a well known technique in the optimization community. Our contribution is to adapt it for the GMKL problem. Towards this end, we use a non-monotone line search criterion which allows the objective function to increase once in a while rather than strictly decrease at each step. This allows us to accept initial step size proposals based on the spectral step length, in many cases even the first, whereas gradient descent would need to try out many step sizes before accepting one that satisfies the Armijo rule. SPG steps are thus much quicker as they require fewer objective function and SVM evaluations.

We also engineer SPG to be more robust to the objective function and gradient noise encountered in large scale GMKL problems. Gradient descent will often get stuck due to inaccurate gradient computation. Only miniscule step sizes will satisfy the Armijo rule since the gradient is pointing in the wrong direction and gradient descent will either time out or take many small steps before recovering. SPG, on the other hand, can recover much faster due to the non-monotone line search criterion. We further exploit this property by deliberately computing objective function values and gradients at low precision when we are far away from the optimum. This allows our initial steps to be much quicker than gradient descent's. As the optimum is approached, we automatically increase the precision of the objective function and gradient computation according to a schedule based on the duality gap and the projected gradient. In many cases, due to this effective scheduling, we need to solve SVMs with a precision of only 10^{-1} at the beginning and at most 10^{-3} towards the end. On the other hand, we found that projected gradient descent needed to solve SVMs with a precision of 10^{-6} from the very start in order to converge.

We carried out extensive experiments comparing the performance of our proposed SPG optimizer to projected gradient descent . We observed that SPG was much faster on even small data sets, optimizing GMKL in a matter of seconds, while gradient descent often struggled to converge. Furthermore, SPG scaled well to large data sets where gradient descent based optimization was infeasible. For instance, without using any parallelization, we were able to train on the Sonar data set with a million kernels and on Covertype with over half a million data points. Note that Covertype is one of the largest data sets on which a single kernel SVM can be trained without parallelization and that our training time was much the same as that of a single SVM with a kernel defined to be the uniformly weighted sum of the given base kernels. In contrast, sequential MKL techniques proposed in the literature have mostly reported results on thousands of kernels or data points (with the exception of SMO-MKL [31]). Finally, in some cases, SPG even outperformed state-of-the-art specialized optimizers designed for a specific MKL formulation. Thus, not only can SPG handle diverse MKL formulations it is also the most efficient optimizer for many of them.

We have implemented SPG in C and use LibSVM [6] as the inner SVM solver. Our implementation builds on top of the LibSVM code base. Our code is easy to modify and new kernel parameterizations and regularizers can readily be plugged in. We therefore expect the code to be useful for trying out new MKL formulations as well as optimizing existing ones. Our objective is to provide a flexible and efficient tool for kernel learning and the SPG-GMKL code can be downloaded from [17].

2. RELATED WORK

Research in Multiple Kernel Learning has focused on both developing new MKL formulations as well as their optimization. Different formulations are required to address the needs of different applications. Early work focused on learning the kernel as a linear combination of given base kernels subject to l_1 or l_2 regularization [12, 19]. This was later extended to handle any $l_{p>1}$ regularizer [18], mixed block regularizers [1] as well as radius-margin based regularization [14]. Non-linear kernel combinations [11, 28], such as products of kernels and mixtures of polynomials, have also been shown to be appropriate in certain domains. Other formulations include multi-class MKL [34], learning over exponentially large kernel combinations [2], learning a different kernel combination per data point [15,20] and MKL discriminant and subspace analysis [9, 32]. Many of these formulations can be easily cast in the GMKL framework.

As regards optimization, specialized algorithms have been developed for many of these formulations. For instance, linear MKL subject to l_1 regularization has been optimized via semi-definite programming [19], M.-Y. regularization [3], semi-infinite linear programming [26] and mirror descent [1]. Sparse MKL models have also been learnt by direct, greedy minimization of the l_0 norm [25]. Linear MKL with l_p regularization has been optimized using the SMO algorithm [31], stochastic gradient descent [21] and semi-infinite linear programming [18]. Stochastic gradient descent has also been used for optimizing multi-class MKL [21, 22] and second order cone programming [27] for learning hyper-kernels.

Unfortunately, most of these specialized optimization techniques are limited to their own formulation and do not generalize well. Gradient descent, on the other hand, can be used to optimize many formulations. It has been used to train linear MKL with l_1 regularization [8,23,29], multi-class

MKL with shared parameters [23], for learning local kernel combinations per data point [15], for hierarchical kernel learning over exponentially large kernel combinations [2] and for learning non-linear kernel combinations [28]. All of these formulations, even if they are not covered under GMKL, can benefit by replacing their gradient descent based optimizer by our proposed SPG algorithm.

3. GMKL: GENERALIZED MULTIPLE KERNEL LEARNING

We review the GMKL formulation in this section and discuss how it can be optimized via gradient descent. We focus on the binary classification problem though other convex loss functions can be substituted as desired.

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 goal in training an SVM 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 $\mathbf{d}$. The primal formulation of GMKL is

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

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. The constraint $\mathbf{d} \geq 0$ can be strengthened or relaxed depending on the cost of projecting back into the feasible set so as to ensure that the parameterized kernel remains positive definite. Allowing the regularizer, kernel and loss function to have such general forms permits GMKL the flexibility to directly model, or be easily extended to, most of the formulations discussed in Section 2. For instance, the kernel can be learnt to be a linear combination of given base kernels, product of kernels or mixture of polynomials, *etc.* Similarly, the regularizer can be set to the 1-norm, any $p > 1$-norm, a mixed block norm, the log determinant of the learnt kernel matrix, *etc.*

The GMKL primal was deliberately formulated as a nested optimization. The key idea is that once the kernel parameters $\mathbf{d}$ are fixed, an efficient, single kernel SVM solver, such as LibSVM, can be used to solve the inner optimization in T while T itself can be optimized via gradient descent.

In order to optimize T using gradient descent, we need to prove that $\boldsymbol{\nabla}_{\mathbf{d}}T$ exists and calculate it efficiently. This is best achieved by looking at T's dual given by

$$\begin{aligned} W(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^t\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t\mathbf{Y}\mathbf{K}_{\mathbf{d}}\mathbf{Y}\boldsymbol{\alpha} + r(\mathbf{d}) \qquad (2) \\ \text{subject to} \quad & \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} = 0, \quad 0 \leq \boldsymbol{\alpha} \leq C \qquad (3) \end{aligned}$$

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 $\boldsymbol{\nabla}_{\mathbf{d}}W$. Proof of the differentiability of W comes from Danskin's Theorem [13]. Since the feasible set is compact, the gradient can be shown to exist if k, r, $\boldsymbol{\nabla}_{\mathbf{d}}k$ and $\boldsymbol{\nabla}_{\mathbf{d}}r$ 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 [8] can be used to show that W have derivatives given by

$$\frac{\partial T}{\partial d_k} = \frac{\partial W}{\partial d_k} = \frac{\partial r}{\partial d_k} - \tfrac{1}{2}\boldsymbol{\alpha}^{*t}\frac{\partial \mathbf{H}}{\partial d_k}\boldsymbol{\alpha}^* \qquad (4)$$

where $\mathbf{H} = \mathbf{Y}\mathbf{K}\mathbf{Y}$. Thus, in order to take a gradient step, all we need to do is obtain $\boldsymbol{\alpha}^*$. Note that since W is equivalent to the single kernel SVM dual with kernel matrix $\mathbf{K}_{\mathbf{d}}$, $\boldsymbol{\alpha}^*$ can be obtained by any SVM optimization package. The projection operator in our case is $\mathbf{P}(\mathbf{d}) = \max(\mathbf{0}, \mathbf{d})$.

To guarantee convergence, the step size s^n is chosen according to the Armijo rule so as to satisfy

$$W(\mathbf{d} + s^n\mathbf{g}) \leq W(\mathbf{d}) + \gamma s^n\mathbf{g}^t\boldsymbol{\nabla}W(\mathbf{d}) \qquad (5)$$

where $\mathbf{g}$ is the chosen direction of descent, the negative gradient $-\boldsymbol{\nabla}W(\mathbf{d})$ in our case, and γ is a tolerance parameter.

4. SPECTRAL PROJECTED GRADIENT DESCENT OPTIMIZATION FOR GMKL

Spectral gradient descent iteratively approximates the objective function with a quadratic model and then optimizes the model at each iteration. It is particularly well suited to large scale problems since it builds a coarse approximation very efficiently and without any memory overhead. Early work by Barzilai and Borwein [4] laid the foundation for choosing the step size as the spectral step length. Grippo's classic non-monotone line search criterion for Newton methods [16] was incorporated into the algorithm by Raydan in [24]. The technique was extended to handle constrained optimization problems in [5] who recommended descending according to the spectral projected gradient rather than the regular gradient so as to reduce the number of projection operations. We refer to this algorithm as Spectral Projected Gradient (SPG) descent and efficiently adapt it for optimizing GMKL based on gradient precision tuning. In addition, we replace the classic non-monotone line search criterion by the one proposed in [33] based on averaging. Proofs of convergence for each of the methods can be found in their respective publications. A few attempts have been made to use SPG for optimizing machine learning problems [10]. However, SPG appears not to be a widely used technique in machine learning or data mining.

Our SPG-GMKL pseudo code is given in Algorithm 1. The three main components are the spectral step length and the spectral projected gradient computation, the non-monotone line search and the SVM solver precision tuning.

Algorithm 1 SPG-GMKL

$n \leftarrow 0$
Initialize $\mathbf{d}^0$ randomly
repeat
  $\boldsymbol{\alpha}^* \leftarrow \text{SolveSVM}_\epsilon(\mathbf{K}(\mathbf{d}^n))$
  $\lambda \leftarrow \text{SpectralStepLength}$
  $\mathbf{p}^n \leftarrow \mathbf{d}^n - \mathbf{P}(\mathbf{d}^n - \lambda\nabla\mathbf{W}(\mathbf{d}^n))$
  $s^n \leftarrow \text{Non-Monotone}$
  $\epsilon \leftarrow \text{TuneSVMPrecision}$
  $\mathbf{d}^{n+1} \leftarrow \mathbf{d}^n - s^n\mathbf{p}^n$
until converged

We discuss these components in detail and also mention our convergence criterion and kernel caching strategy.

Spectral step length and projected gradient.

We describe how to move from the current iterate $\mathbf{d}^n$ to the next iterate $\mathbf{d}^{n+1}$. The step size and the descent direction are chosen so as to reduce the number of SVM evaluations and projection operations as compared to regular projected gradient descent.

SPG-GMKL chooses it's step size based on second order information. At each iteration, it approximates the GMKL objective function $W(\mathbf{d})$ by a quadratic in $\mathbf{d}$. The step size is then based on the eigenvalues of the approximated Hessian. A particularly simple choice of the Hessian, $\lambda^{-1}\mathbf{I}$, is made so as to be able to efficiently scale to large problems. Imposing the constraint that the directional derivatives of the objective and the approximated functions match at the current point leads to the following choice for λ

$$\lambda^n = \frac{\left\langle \mathbf{d}^n - \mathbf{d}^{n-1}, \mathbf{d}^n - \mathbf{d}^{n-1} \right\rangle}{\left\langle \mathbf{d}^n - \mathbf{d}^{n-1}, \boldsymbol{\nabla} W(\mathbf{d}^n) - \boldsymbol{\nabla} W(\mathbf{d}^{n-1}) \right\rangle} \tag{6}$$

If λ^n is negative then it is set to λ_{Max}, else it is clipped between $\lambda_{\text{Min}} = 10^{-30}$ and $\lambda_{\text{Max}} = 10$ so as to ensure that the steps are always bounded.

Standard methods would have taken a projected gradient step of size λ^n. However, in SPG, we instead step in the direction of the negative spectral projected gradient. In particular, the next iterate is obtained as

$$\mathbf{d}^{n+1} = \mathbf{d}^n - s^n \mathbf{p}^n \tag{7}$$

$$\text{where} \quad \mathbf{p}^n = \mathbf{d}^n - \mathbf{P}(d^n - \lambda^n \boldsymbol{\nabla} W(\mathbf{d}^n)) \tag{8}$$

where s^n is selected from $\{1, 1/2, 1/4, \ldots\}$ according to the non-monotone line search criterion, $\mathbf{p}$ is the spectral projected gradient and $\mathbf{P}$ is the projection operator in our case $\mathbf{P}(\mathbf{d}) = \max(\mathbf{d}, \mathbf{0})$. This choice of descent direction ensures that the projection operator needs to be applied only once per iteration. While the cost of projection is negligible in our case it might become significant if the constraints $\mathbf{d} \geq 0$ are replaced by more complicated ones.

Non-monotone line search criterion.

We employ a non-monotone line search criterion so as to reduce the number of SVM evaluations and be robust to the objective function and gradient noise inherent in GMKL optimization. The classical SPG non-monotone line search criterion selects step sizes s^n that satisfy

$$W(\mathbf{d}^n - s^n \mathbf{p}^n) \leq \max_{0 \leq j \leq M} W(\mathbf{d}^{n-j}) - \gamma s^n \boldsymbol{\nabla}^t W(\mathbf{d}^n) \mathbf{p}^n \tag{9}$$

So as to allow the objective function to increase once in a while. However, [33] observed that, for unconstrained quasi-Newton methods, maximizing over the previous function values could be unstable in some cases and that averaging could be a much better alternative. We therefore incorporate their criterion into SPG-GMKL by choosing s^n satisfying

$$W(\mathbf{d}^n - s^n \mathbf{p}^n) \leq C^n - \gamma s^n \boldsymbol{\nabla}^t W(\mathbf{d}^n) \mathbf{p}^n \tag{10}$$

$$Q^{n+1} = \eta^n Q^n + 1 \tag{11}$$

$$C^{n+1} = (\eta^n Q^n C^n + \mathbf{W}(\mathbf{d}^{n+1}))/Q^{n+1} \tag{12}$$

where we set $C^0 = W(\mathbf{d}^0)$, $Q^0 = 1$ and $\gamma = 10^{-4}$. The parameter η influences the number of iterations over which W is averaged. Setting $\eta = 0$ reduces the criterion to the Armijo rule while setting $\eta = 1$ implies averaging W over all previous iterations. We follow the heuristic of [33] and vary η dynamically depending on the behavior of the objective function and the approximated quadratic. If the approximation is good (bad) then we increase (decrease) η by 0.025 making sure to clip it between $[0.1, 1]$ else leave it unchanged.

Tuning the tolerance of the SVM solver.

The SPG-GMKL optimizer is robust to imprecisions in calculating the objective function and the gradient. We exploit this by solving SVMs with a tolerance of only 10^{-1} in the initial iterations. This is automatically decreased as we move closer to the optimum or the step size becomes too small. The SVM solver's tolerance is set as

$$\epsilon = \begin{cases} \min(\epsilon, 10^{-1}) & \text{if } (1 \leq u) \text{ or } (5 \leq v) \\ \min(\epsilon, 10^{-2}) & \text{if } (0.1 \leq u < 1) \text{ or } (1 \leq v < 5) \\ \min(\epsilon, 10^{-3}) & \text{if } (u < 0.1) \text{ or } (v < 1) \\ \max(\epsilon/10, 10^{-5}) & \text{if } s^n < 10^{-8} \end{cases} \tag{13}$$

where u is the duality gap (whenever available) and v is the projected gradient norm.

Convergence Criterion and Kernel Caching.

We declare the SPG-GMKL algorithm to have converged whenever the duality gap reduces below 10^{-3} or, if the duality gap is unavailable, whenever the l_2 or l_∞ norm of the projected gradient reduces below 0.04. The reason the stopping criterion is so loose in terms of the projected gradient norm is because we do not assume that our inner SVM solver produces high quality solutions, and hence there is significant noise in the gradient estimates. In fact, the condition on the projected gradient seems stricter since, in many of the small data sets, the duality gap fell below 10^{-3} before the projected gradient norm reduced below 0.04. However, on some of the larger data sets, the projected gradient norm was found to become small even when the duality gap was large. We therefore use the duality gap as a stable stopping criterion whenever it is available (for sum of kernels in our experiments).

Kernel caching strategies can have a big impact on performance since kernel computations can dominate everything else in some cases. While a few different kernel caching techniques have been explored for SVMs, we stick to the standard one used in LibSVM [6]. A Least Recently Used (LRU) cache is implemented as a circular queue. Each element in the queue is a pointer to a recently accessed (common) row of each of the individual kernel matrices.

Advantages over projected gradient descent.

Our SPG-GMKL implementation has a number of advantages over regular projected gradient descent. First, we are able to efficiently bring second order information into play by choosing our step size according to the spectral step length. This was found to be very beneficial for many data sets. Second, we can significantly reduce the number of SVM evaluations by combining the spectral step length computation with a non-monotone line search criterion. While gradient descent might take ten or more SVM evaluations to find a step size satisfying the Armijo rule, SPG-GMKL will frequently take only a single SVM evaluation as the first step size proposal is accepted. The non-monotone line

search might also have beneficial side effects. Non-convex kernel parameterizations might benefit as such criteria have been shown to sometimes help escape poor local optima. Furthermore, non-monotonicity can also be helpful in cases when the objective function starts resembling a long, narrow valley where standard gradient descent techniques can get stuck. Third, our SPG-GMKL implementation is robust to noise in the calculation of the gradient and the objective function. Gradient descent can require hundreds of SVM evaluations in a line search when the gradient is not pointing in the right direction. SPG-GMKL is typically more robust and can recover much faster. Fourth, by starting with a low SVM precision and then dynamically increasing it as required, we can make the cost of evaluating each SVM much lower. This was found to give an order of magnitude improvement over gradient descent on some large data sets where the cost of evaluating an SVM is very high.

5. EXPERIMENTS

We carry out extensive experiments to test the performance of SPG-GMKL along various dimensions. We benchmark its performance on small, medium and large scale data sets. We show that SPG-GMKL consistently outperforms projected gradient descent by a wide margin and that SPG-GMKL can often be more than ten times faster. We also demonstrate that SPG-GMKL can scale well to large problems involving a million kernels on Sonar or over half a million data points on Covertype. This is particularly significant as Covertype is one of the largest data sets on which an RBF SVM can be trained without parallelization.

We also optimize four different MKL formulations to demonstrate the flexibility of SPG-GMKL. We learn two different types of kernel combinations – sum of kernels with $K = \sum_k d_k K_k$ and product of kernels with $K = \prod_k K_k(d_k) = \prod_k e^{-d_k D_k}$ corresponding to tuning the RBF bandwidth while combining distance functions D_k. We regularize these with the l_1 and $l_{p>1}$ norms. We focus on the linear MKL formulations in Subsections 5.1 and 5.2. Specialized optimizers have been developed for learning a sum of base kernels subject to l_1 regularization as it is one of the most popular formulations. We show that the performance of general purpose SPG-GMKL can be much better than state-of-the-art implementations of some of these specialized optimizers. We then use SPG-GMKL to learn products of kernels in Subsections 5.3 and 5.4 which cannot be optimized using any other solver (apart from projected gradient descent, of course). We show that SPG-GMKL scales well to many problems where projected gradient descent fails to converge.

Finally, in Subsection 5.5, we study the scaling properties of SPG-GMKL and the contribution of each of the individual components. On the data sets that we tried, SPG-GMKL exhibited a linear dependence on the number of kernels and a sub-quadratic dependence on the number of training points. We also show that each of the three major components of SPG-GMKL, namely the spectral step length, the non-monotone line search and the SVM solver precision tuning, are necessary and that removing any of them can lead to substantial degradations in performance.

We briefly describe the various implementations that we tested. We refer to our algorithm as SPG-GMKL and the projected gradient descent algorithm as PGD. The SPG-GMKL implementation is obtained by starting from the PGD codebase and adding spectral step length computation, the non-monotone line search criterion and SVM solver tolerance tuning. This ensures an absolutely fair comparison between the two implementations. Another projected gradient descent implementation is available in the form of SimpleMKL [23]. The code was downloaded from the authors' website. However, this implementation is meant only for linear MKL subject to l_1 regularization and hence we compare to it only in Subsection 5.2. For this formulation, a very efficient implementation of the SILP-MKL [26] optimizer is available in the Shogun toolbox. We downloaded the latest version of Shogun (v 0.9.3) and used the highly optimized CPLEX solver to solve their outer loop LP. Note that the linear MKL formulations can be made convex with a simple change of variables. As such, SPG-GMKL and PGD converge to the same solution. SimpleMKL and SILP-MKL can also be made to converge to the same solution with an appropriate choice of parameters. We therefore report only the training time in our experiments as the objective function value, number of kernels selected and prediction accuracy on the test set are similar for all the methods. SPG-GMKL and PGD might potentially converge to different solutions for the non-convex formulations involving products of kernels. We did not observe this behavior though, and both implementations returned very similar solutions. Table 1 summarizes the statistics of the data sets that were tried. For linear MKL, kernels were generated as recommended in [23] for the small data sets. We generated RBF kernels with ten bandwidths for each individual dimension of the feature vector as well as the full feature vector itself. Similarly, we also generated polynomial kernels of degrees 1, 2 and 3. Kernel matrices were pre-computed and normalized to have unit trace. A fixed number of kernels were computed on the fly for the medium and large data sets. All kernels were computed on the fly for products of kernels where we defined one RBF kernel per feature for the small data sets and a fixed number for the large data sets. We perform 5 fold cross validation on the small data sets. This gives both the mean training time and its variance so as to better judge

Table 1: Data set statistics

Data sets	# Train	# Dim	# Kernels	
			Sum	Product
Leukemia	38	7129		7129
Wpbc	194	34	455	34
Sonar	208	60	793	60
Liver	345	6	91	6
Ionosphere	351	34	455	34
Breast-Cancer	683	10	143	10
Australian	690	14	195	14
Diabetes	768	8	117	8
Letter	20000	16		16
RCV1	20242	47236		50
Adult-8	22696	123	50	42
Web-7	24692	300	50	43
Poker	25010	10		10
Adult-9	32561	123	50	42
Web-8	49749	300	50	43
KDDCup04-Physics	50000	71	50	
Cod-RNA	59535	8	50	8
Real-Sim	72309	20958	25	
Covertype	581012	54	5	

Table 2: $l_{p>1}$-SumMKL results on small data sets.

	$p = 1.1$		$p = 1.33$		$p = 2.0$	
Data sets	PGD (s)	SPG-GMKL (s)	PGD (s)	SPG-GMKL (s)	PGD (s)	SPG-GMKL (s)
Australian	39.4 ± 6.0	7.0 ± 1.6	248.3 ± 329.7	6.9 ± 1.2	23.5 ± 2.3	6.0 ± 0.2
Liver	282.2 ± 226.0	0.8 ± 0.1	3.3 ± 2.5	0.7 ± 0.1	0.7 ± 0.1	0.6 ± 0.1
Sonar	785.5 ± 471.1	9.0 ± 1.8	57.8 ± 39.8	4.2 ± 0.4	7.7 ± 1.1	3.1 ± 0.3
Breast-Cancer	237.2 ± 97.8	8.6 ± 2.2	11.2 ± 2.4	3.3 ± 0.1	6.6 ± 0.6	2.7 ± 0.1
Diabetes	73.6 ± 38.8	4.1 ± 0.5	11.5 ± 2.4	3.0 ± 0.1	3.6 ± 0.1	2.5 ± 0.1
Wpbc	44.4 ± 11.6	1.2 ± 0.4	5.3 ± 1.8	1.3 ± 0.5	1.7 ± 0.2	1.2 ± 0.1

Table 3: $l_{p=1.33}$-SumMKL results on large data sets

Data sets	PGD (hrs)	SPG-GMKL (hrs)
Adult-9	31.77	4.42
Cod-RNA	66.48	19.10
KDDCup04	-	42.20
Covertype	-	64.46
Sonar1M	-	105.62

Table 6: l_1-SumMKL results on large data sets

Data sets	PGD (hrs)	SPG-GMKL (hrs)
Adult-9	35.84	4.55
Cod-RNA	-	25.17
KDDCup04	-	40.10
Real-Sim	-	45.94

the significance of our results. The SVM parameter C was chosen by validation so as to maximize the prediction accuracy. All experiments were carried out on a single core of a 2.1 Ghz AMD 6172 processor with 48 Gb RAM. The results on Covertype were obtained on a similar machine with only 14 Gb RAM.

5.1 Sum of kernels with $l_{p>1}$ regularization

We learn a linear combination of base kernels subject to $l_{p>1}$ regularization. This formulation has yielded state-of-the-art results [22] for combining multiple, heterogeneous features for recognizing objects on the challenging Caltech image data sets. We focus on comparing SPG-GMKL to PGD in this Subsection. Table 2 summarizes the results on the small data sets. For small values of p, the optimization problem is hard, and SPG-GMKL can be two orders of magnitude faster than PGD. As the value of p increases, the optimization problem becomes much simpler, and the two techniques become more comparable though SPG-GMKL can still be twice as fast as PGD.

Results on the larger data sets are given in Table 3. PGD is again much slower and fails to converge on KDDCup04-Physics with fifty thousand points, Covertype with nearly six hundred thousand points and Sonar with a million kernels. SPG-GMKL, without any parallelization, converged on Covertype with 26 SVM evaluations in 64 hours which is reasonable given that solving just the first SVM, equivalent to solving a single kernel SVM obtained by linearly summing the five base kernels with uniform weights, took 44 hours. Note that we were caching only 0.19% of the support vectors and thus the training time can be considerably speeded up by utilizing a larger kernel cache. Similar trends were observed for Sonar with a million kernels.

Figure 1 plots the decrease in objective value versus time on a log-log scale. SPG-GMKL's initial steps are much quicker than PGD's due to the combination of the spectral step length, the non-monotone line search and the coarse SVM precision. SPG-GMKL therefore gets close, and converges, to the optimum much faster than PGD. The non-monotone behavior of SPG-GMKL is prominent on both the Web and Sonar20K datasets. Note that, on Web, the objective function seems to dip below the global minimum. This is due to noise in the objective function because of the coarse SVM precision. PGD would have gotten stuck here as it insists on a strict decrease in the objective function at each iteration. SPG-GMKL can recover due to its non-monotone criterion and SVM precision tuning. This demonstrates that SPG-GMKL is more robust to noise than PGD.

5.2 Sum of kernels with l_1 regularization

One of the most popular MKL formulations is to learn a sum of given base kernels subject to l_1 regularization on the combination coefficients. The joint winner of the highly competitive PASCAL VOC object detection challenge [30] employed such a formulation to learn a sparse combination of features for efficient object detection in images. A number of optimizers have been developed for this formulation. The state-of-the-art is defined by SILP-MKL and SimpleMKL. Table 4 shows that SPG-MKL can not only outperform PGD on this formulation, it is also much better than SILP-MKL and SimpleMKL. For instance, SPG-GMKL is always more than twice as fast as SILP-MKL and can sometimes be orders of magnitude faster as the number of kernels is increased. This demonstrates that our general purpose SPG-GMKL solver can be more efficient than state-of-the-art optimizers designed for a specific formulation. SPG-GMKL is also an order of magnitude faster than SimpleMKL and PGD which are two different implementations of the projected gradient descent algorithm. HessianMKL [7] provides a second order method for optimizing this formulation. While we found HessianMKL to be an improvement over SimpleMKL on small data sets, it's performance was still much worse than that of SPG-GMKL.

5.3 Product of kernels with $l_{p>1}$ regularization

We demonstrate the flexibility of SPG-GMKL by using it to learn products of kernels subject to $l_{p>1}$ regularization. Note that this formulation presents a challenging non-convex optimization problem. Tables 5 and 8 detail performance on small and large data sets respectively. PGD failed to converge in many cases and, when it did, it was often ten to a hundred times slower than SPG-GMKL. We found that the main reason for PGD not converging was that its step size was approaching zero due to inaccurate gradient computation coupled with the Armijo rule. As a result, PGD required many SVM evaluations to take a miniscule step and timed out after a considerable period. On the other hand,

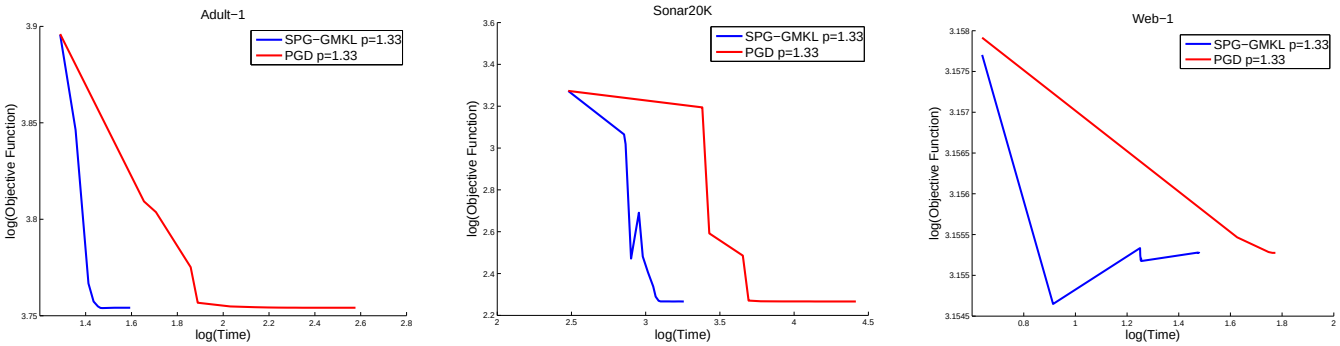

Figure 1: Decrease in objective value with time on a log-log scale for SPG-GMKL and PGD.

Table 4: l_1-SumMKL results on small data sets.

Data sets	SimpleMKL (s)	SILP-MKL (s)	PGD (s)	SPG-GMKL (s)
Wpbc	400.2 ± 128.4	15.5 ± 7.7	38.2 ± 17.6	6.2 ± 4.2
Breast-Cancer	676.1 ± 356.4	12.3 ± 1.2	57.9 ± 85.1	5.1 ± 0.6
Australian	383.2 ± 33.5	1094.9 ± 621.6	29.5 ± 7.1	10.1 ± 0.8
Ionosphere	1247.6 ± 680.0	107.8 ± 18.8	1392.4 ± 824.2	39.2 ± 6.8
Sonar	1468.2 ± 1252.7	935.1 ± 65.0	–	273.4 ± 64.0

Table 5: $l_{p>1}$-ProdMKL results on small data sets.

	$p = 1.1$		$p = 1.33$		$p = 2.0$	
Data sets	PGD (s)	SPG-GMKL (s)	PGD (s)	SPG-GMKL (s)	PGD (s)	SPG-GMKL (s)
Australian	–	22.0 ± 8.4	–	22.1 ± 6.7	–	21.6 ± 6.4
Liver	722.4 ± 880.0	1.0 ± 0.3	136.1 ± 167.9	0.6 ± 0.2	0.3 ± 0.1	0.4 ± 0.1
Sonar	–	7.4 ± 2.9	–	3.2 ± 0.5	293.8 ± 574.7	3.1 ± 0.5
Breast-Cancer	28.4 ± 42.5	4.0 ± 1.0	448.6 ± 691.5	3.3 ± 0.8	377.8 ± 712.0	3.2 ± 0.5
Diabetes	1232.5 ± 716.2	4.8 ± 2.4	364.4 ± 718.3	2.5 ± 0.5	2.1 ± 0.4	2.1 ± 0.1
Wpbc	1080.3 ± 90.9	4.4 ± 1.8	1064.7 ± 41.2	3.7 ± 1.4	209.3 ± 390.8	2.1 ± 0.4

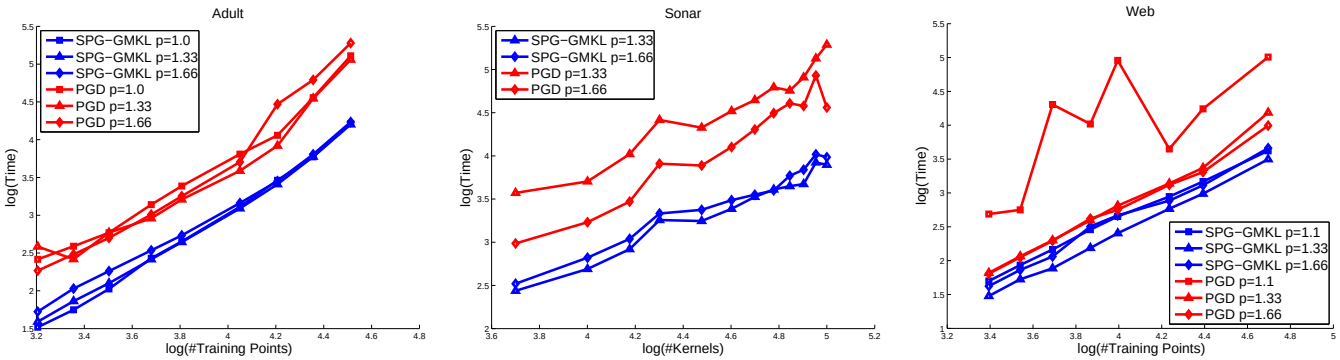

Figure 2: Variation in training time with the number of kernels and data points for SPG-GMKL and PGD.

Table 7: Effects of individual components on small data sets.

Data sets	PGD		PGD+N		PGD+S		PGD+N+S	
	Time (s)	# SVMs	Time (s)	# SVMs	Time (s)	# SVMs	Time (s)	# SVMs
Australian	39.4 ± 6.0	3230	32.7 ± 3.6	116	317.0 ± 49.1	5980	7.0 ± 1.6	621
Sonar	785.5 ± 471.1	209461	41.6 ± 17.1	3236	40.2 ± 24.6	3806	9.0 ± 1.8	2457
Breast-Cancer	237.2 ± 97.8	109599	42.2 ± 4.1	1187	14.9 ± 2.2	3537	8.6 ± 2.2	3006
Diabetes	73.6 ± 38.8	29347	26.3 ± 9.5	2966	10.5 ± 2.6	1239	4.1 ± 0.5	695
Wpbc	44.4 ± 11.6	14376	27.9 ± 13.6	9388	2.9 ± 0.8	340	1.2 ± 0.4	79
Adult-1	376	196	356	196	104	118	71	50
Adult-2	547	238	549	238	207	115	143	70

Table 8: $l_{p=1.33}$-ProdMKL results on large data sets.

Data sets	PGD (hrs)	SPG-GMKL (hrs)
Letter	18.89	0.66
Poker	2.29	0.96
Adult-8	-	3.42
Web-7	-	1.33
RCV1	-	15.93
Cod-RNA	-	8.99

SPG-GMKL generally required only a single SVM evaluation per iteration and converged quickly to the optimum.

5.4 Product of kernels with l_1 regularization

We finally use SPG-GMKL to learn products of kernels subject to l_1 regularization. This formulation was observed to be very good for feature selection [28] as it was significantly better than leading filter and wrapper methods. Tables 9 and 10 compare the performance of SPG-GMKL and PGD on small and large data sets respectively. The trends, and explanations, are much the same as for l_p regularized product of kernels but SPG-GMKL's gains over PGD are even bigger. For instance. On many of the small data sets, SPG-GMKL was a thousand times faster than PGD.

5.5 Scaling Properties and component contributions

We now investigate how SPG-GMKL scales with the number of kernels and data points. Figure 2 plots the results for linear MKL with various values of p on a log-log scale. The number of training points is varied on the Adult and Web data sets while the number of kernels is varied on Sonar. We estimate, from the plots, that SPG-GMKL, on average, scales linearly with the number of kernels on Sonar and as $n^{1.98}$ and $n^{1.52}$ with the number of training points on Adult and Web. The scaling factors are very similar for PGD though the constants involved are much bigger.

We also determine the contribution of each of the three major components of SPG-GMKL, namely the spectral step length, the non-monotone line search and the SVM precision tuning. In Table 7, we start with the PGD implementation as the baseline. We then turn on either the spectral step length computation (PGD+S), or the non-monotone line search (PGD+N), or both (PGD+N+S) and assess the impact on both training time and the number of SVM evaluations. As can be seen, adding either component in isolation does not lead to the best gains and, in fact, can sometimes degrade performance below the PGD baseline. It is only when both the components are combined that one consistently gets improvements in performance.

The full impact of tuning the SVM precision is best seen on larger data sets where the cost of evaluating an SVM is

Table 9: l_1-ProdMKL results on small data sets.

Data sets	PGD (s)	SPG-GMKL (s)
Australian	–	16.6 ± 7.2
Sonar	–	4.8 ± 1.7
Liver	1482.6 ± 635.7	2.6 ± 2.0
Breast-Cancer	1182.6 ± 662.4	4.4 ± 1.5
Diabetes	1525.7 ± 552.6	7.6 ± 6.6
Wpbc	1093.9 ± 187.5	2.7 ± 1.0
Leukemia	2961.5 ± 0.0	850.0 ± 0.0

Table 10: l_1-ProdMKL results on large data sets.

Data sets	PGD (hrs)	SPG-GMKL (hrs)
Letter	18.66	0.67
Poker	5.57	0.49
Adult-8	-	1.73
Web-7	-	0.88
RCV1	-	18.17
Cod-RNA	-	3.45

Table 11: Effects of individual components on large data sets

Data sets	PGD (hrs)	PGD+N+S (hrs)	SPG-GMKL (hrs)
Adult-9	31.77	8.33	4.43
Web-8	4.27	1.73	0.87
Sonar100K	53.91	3.35	2.19

significant. Table 11 shows that adding spectral step length computation and the non-monotone line search criterion to the PGD baseline cuts down its training time to at least half on Web and by more than fifteen times on Sonar. SPG-GMKL adds SVM precision tuning to the combination to further reduce the training time by a factor of two. Thus, each of the three major components that we proposed must be present in order for SPG-GMKL to be an efficient solver that scales well to large problems.

6. CONCLUSIONS

In this paper, we developed an efficient, spectral projected gradient descent based optimizer for the Generalized Multiple Kernel Learning framework. Starting from the projected gradient descent algorithm, we added three components based on the spectral step length, the non-monotone line search criterion and SVM precision tuning. We demonstrated that each of the components is necessary to the success of the SPG-GMKL optimizer.

We carried out extensive experiments benchmarking the performance of SPG-GMKL. We showed that SPG-GMKL routinely gives order of magnitude improvements over PGD and scales well to problems where PGD does not converge. For instance, SPG-GMKL was able to train on problems with a million kernels or half a million data points. This redefines the state-of-the-art in kernel learning optimization. Furthermore, SPG-GMKL, even though it is a general purpose solver, was able to outperform leading specialized solvers developed for a specific formulation. As such, SPG-GMKL is one of the most efficient techniques for kernel learning.

We tested the flexibility of SPG-GMKL by having it optimize four different MKL formulations. Our code is easy to modify and new formulations can readily be plugged in. Our objective is to provide an efficient tool for rapidly prototyping new MKL formulations and trying out existing ones on real world applications.

Acknowledgements

We are grateful to Kamal Gupta and to the Computer Services Center at IIT Delhi.

7. REFERENCES

[1] J. Aflalo, A. Ben-Tal, C. Bhattacharyya, J. S. Nath, and S. Raman. Variable sparsity kernel learning. *JMLR*, 12:565–592, 2011.

[2] F. R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, pages 105–112, 2008.

[3] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, pages 6–13, 2004.

[4] J. Barzilai and J. Borwein. Two-point step size gradient methods. *IMA J. on Numerical Analysis*, 8:141–148, 1988.

[5] E. G. Birgin, J. M. Martinez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. on Optimization*, 10(4):1196–1211, 2000.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] O. Chapelle and A. Rakotomamonjy. Second order optimization of kernel parameters. In *NIPS Workshop on Automatic Selection of Optimal Kernels*, 2008.

[8] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for Support Vector Machines. *Machine Learning*, 46:131–159, 2002.

[9] J. Chen, S. Ji, B. Ceran, Q. Li, M. Wu, and J. Ye. Learning subspace kernels for classification. In *KDD*, pages 106–114, 2008.

[10] D. Cores, R. Escalante, M. Gonzalez-Lima, and O. Jimenez. On the use of the spectral projected gradient method for support vector machines. *Computational and Applied Mathematics*, 28(3):327–364, 2009.

[11] C. Cortes, M. Mohri, and A. Rostamizadeh. Learning non-linear combinations of kernels. In *NIPS*, pages 396–404, 2009.

[12] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.

[13] J. M. Danskin. *The Theorey of Max-Min and its Applications to Weapons Allocation Problems*. 1967.

[14] K. Gai, G. Chen, and C. Zhang. Learning kernels with radiuses of minimum enclosing balls. In *NIPS*, pages 649–657, 2010.

[15] M. Gonen and E. Alpaydin. Localized multiple kernel learning. In *ICML*, pages 352–359, 2008.

[16] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for newton's method. *SIAM J. on Numerical Analysis*, 23:707–716, 1986.

[17] A. Jain, S. V. N. Vishwanathan, and M. Varma, 2012. The SPG-GMKL code `http://research.microsoft.com/~manik/code/SPG-GMKL/download.html`.

[18] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Muller, and A. Zien. Efficient and accurate l_p-norm Multiple Kernel Learning. In *NIPS*, pages 997–1005, 2009.

[19] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

[20] C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *JMLR*, 6:1043–1071, 2005.

[21] F. Orabona and L. Jie. Ultra-fast optimization algorithm for sparse multi kernel learning. In *ICML*, pages 249–256, June 2011.

[22] F. Orabona, L. Jie, and B. Caputo. Online-batch strongly convex multi kernel learning. In *CVPR*, pages 787–794, San Francisco, California, June 2010.

[23] A. Rakotomamonjy, F. Bach, Y. Grandvalet, and S. Canu. SimpleMKL. *JMLR*, 9:2491–2521, 2008.

[24] M. Raydan. The barzilai and borwein gradient method for the large scale unconstrained minimization problem. *SIAM J. on Optimization*, 7(1):26–33, 1997.

[25] V. Sindhwani and A. C. Lozano. Non-parametric group orthogonal matching pursuit for sparse learning with multiple kernels. In *NIPS*, pages 2519–2527, 2011.

[26] S. Sonnenburg, G. Raetsch, C. Schaefer, and B. Schoelkopf. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2006.

[27] I. W. Tsang and J. T. Kwok. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17(1):48–58, 2006.

[28] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, page 134, 2009.

[29] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, pages 1–8, 2007.

[30] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, pages 606–613, 2009.

[31] S. V. N. Vishwanathan, Z. Sun, N. Theera-Ampornpunt, and M. Varma. Multiple kernel learning and the SMO algorithm. In *NIPS*, pages 2361–2369, 2010.

[32] J. Ye, , S. Ji, and J. Chen. Multi-class discriminant kernel learning via convex programming. *JMLR*, 9:719–758, 2008.

[33] H. Zhang and W. W. Hager. A nonmonotone line search technique and its application to unconstrained optimization. *SIAM J. on Optimization*, 14(4):1043–1056, 2004.

[34] A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *ICML*, pages 1191–1198, 2007.