Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. TR-178

Infinite Kernel Learning

Peter Vincent Gehler and Sebastian Nowozin[1]

October 2008

[1] Department for Empirical Inference, email: peter.gehler;sebastian.nowozin@tuebingen.mpg.de

Infinite Kernel Learning

Peter Vincent Gehler and Sebastian Nowozin

Abstract. In this paper we consider the problem of automatically learning the kernel from general kernel classes. Specifically we build upon the Multiple Kernel Learning (MKL) framework and in particular on the work of (Argyriou, Hauser, Micchelli, & Pontil, 2006). We will formulate a Semi-Infinite Program (SIP) to solve the problem and devise a new algorithm to solve it (Infinite Kernel Learning, IKL). The IKL algorithm is applicable to both the finite and infinite case and we find it to be faster and more stable than SimpleMKL (Rakotomamonjy, Bach, Canu, & Grandvalet, 2007) for cases of many kernels. In the second part we present the first large scale comparison of SVMs to MKL on a variety of benchmark datasets, also comparing IKL. The results show two things: a) for many datasets there is no benefit in linearly combining kernels with MKL/IKL instead of the SVM classifier, thus the flexibility of using more than one kernel seems to be of no use, b) on some datasets IKL yields impressive increases in accuracy over SVM/MKL due to the possibility of using a largely increased kernel set. In those cases, IKL remains practical, whereas both cross-validation or standard MKL is infeasible.

---

1 Introduction

In this work we consider the task of binary classification with a Support Vector Machine (SVM). Assume a set of training points $S = \{(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)\}$ is given, with samples $x_i \in \mathcal{X}$ and labels $y_i \in \{-1, +1\}$. A classification function $f : \mathcal{X} \to \{-1, +1\}$ is sought which yields small generalization error. The SVM algorithm searches for an hyperplane in a feature space defined by some mapping $\phi : \mathcal{X} \to \mathcal{H}$ via minimization of the following objective $(w^*, b^*) = \arg\min_{w \in \mathcal{H}, b \in \mathbb{R}} \frac{1}{2}\|w\|_{\mathcal{H}} + C \sum_{i=1}^{n} L(y_i, \langle w, \phi(x_i)\rangle_{\mathcal{H}} + b)$. Here L denotes some loss function, i.e. the hinge loss $L(x, t) = \max(0, 1 - xt)$. In the remainder of the paper we will use the so-called kernel trick and replace all dot products $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ with a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The representer theorem (Kimeldorf & Wahba, 1971; Schölkopf, Herbrich, & Smola, 2001) guarantees that an optimal solution of the above minimization problem admits a representation of the form $f(x) = \sum_{i=1}^{n} y_i \alpha_i k(x, x_i) + b$ as expansion around the training samples x_i.

There are two ingredients which have to be specified prior to SVM training: the feature space $\mathcal{H}$ and the strength of regularization. The function class (hyperplanes in $\mathcal{H}$) is determined by means of the kernel function k, where we will use the notation $k(\cdot, \cdot; \theta)$ to express its dependency on some kernel parameters θ. We will think of θ as specifying the type of kernel *and* its corresponding parameters, i.e. Gaussian and the bandwidth or polynomial and the degree. The regularization is controlled by a constant, denoted by $C \in \mathbb{R}^+$, which controls the trade off between smoothness of the prediction function and the ability to explain the training data correctly.

The ideal measure to select these parameters is the generalization error and C and θ should be set such that the generalization error is minimized. Since one does not have access to this measure, upper bounds as the Span-Bound or Radius Margin bound have been proposed to approximate it, see (Chapelle, Vapnik, Bousquet, & Mukherjee, 2002) for an application to SVM learning. Arguably the most prominent estimator is cross-validation (CV) which has the shortcoming of being computationally expensive, especially in the case of many adjustable kernel parameters.

The approach of Multiple Kernel Learning (MKL) (Lanckriet, Cristianini, Bartlett, Ghaoui, & Jordan, 2004; Bach, Lanckriet, & Jordan, 2004; Sonnenburg, Rätsch, Schäfer, & Schölkopf, 2006) follows a different route to select a set of kernel parameters. Here only the parameter C and a set of M so-called *base-* or *proposal-*kernels $k(\cdot, \cdot; \theta_m)$ have to be specified. Within this framework the final kernel is a mixture of the proposal kernels $k(x, x'; \theta) = \sum_{m=1}^{M} d_m k(x, x'; \theta_m)$. In (Lanckriet et al., 2004) no conditions on the kernel parameters are imposed but instead the final matrix is required to be positive definite. This turns the problem into a semi-definite program (SDP) for which the best known algorithms scale in the order $O(n^6)$. One can impose a simplex constraint on the mixing coefficients, turning the final kernel into a convex combination of the proposal kernels. This constraint is also used in this work. The mixing parameters d_m are found simultaneously with the SVM parameters w and b during the minimization of the SVM objective, while the parameters θ_m are kept fixed at all times. Al-

though the new objective function contains product-terms between variables, it is still possible to state the problem as jointly-convex optimization problem (Zien & Ong, 2007; Sonnenburg et al., 2006).

As shown in (Argyriou et al., 2006), keeping the number M of base kernels fixed is an unnecessary restriction and one can instead search over a possibly infinite set of base-kernels, e.g. all Gaussian kernels with separate bandwidths $\sigma_d \in [\sigma_{\min}, \sigma_{\max}]$ for each input dimension d. We will show experimentally that this flexibility of the kernel class can translates into significant improvements in terms of accuracy.

The approach of combining multiple kernels offers the flexibility of combining several heterogeneous data sources in a single framework. It has been applied with much success to the task of object classification (Bosch, Zisserman, & Munoz, 2007) which aims to separate images of different categories of objects like chairs, cars, flowers and trees. For each image a variety of different feature descriptors is available characterizing different aspects of the image, like color or texture. The different descriptors vary in discriminability for different categories an orange for example might better be described by color features, while a car is better represented by shape information. Using MKL by far the best results have been obtained on the Caltech benchmark datasets (Fei-Fei, Fergus, & Perona, 2004; Griffin, Holub, & Perona, 2007) as of now (Bosch et al., 2007). In (Sonnenburg et al., 2006) it is argued that MKL is more interpretable than the ordinary SVM, since it is possible to inspect the selected kernels after training which can reveal properties of the dataset.

This paper is organized as follows. In the next section we will present the proposed approach as the limit of MKL. Section 3 will discuss the most important ingredient of this approach and relate to other work. In Section 4 we present the experiments comparing our approach to existing ones. We conclude with a discussion in Section 5.

2 Infinite Kernel Learning

We adopt the formulation of the primal objective function for MKL from (Zien & Ong, 2007), using the convention $\frac{1}{0} := 0$. Throughout the paper we will use the notation Θ_f to denote finite sets and Θ for sets of arbitrary cardinality.

$$\begin{aligned} \min_{d,v,\xi,b} \quad & \frac{1}{2}\sum_{\theta\in\Theta_f}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i\left(\sum_{\theta\in\Theta_f}\langle v_\theta, \phi_\theta(x_i)\rangle + b\right) \geq 1-\xi_i, \quad i=1,\ldots,n \\ & \xi_i \geq 0, \quad i=1,\ldots,n \\ & \sum_{\theta\in\Theta_f} d_\theta = 1, \qquad d_\theta \geq 0. \end{aligned} \tag{1}$$

This primal formulation uses the technique of substituting $v_\theta := d_\theta w_\theta$ so that the resulting problem becomes convex in both d_θ and v_θ. Without this substitution the problem would be in a more intuitive form but non-convex due to product terms between both variables.

In (Argyriou et al., 2006) it was noted that the set of base kernels needs not to be of finite size but can be an infinite set of kernels. We build upon this insight and adopt it using the standard MKL objective (1). The kernel is optimized over the entire set of kernels parametrized by Θ, namely the convex hull of $\{k(\cdot,\cdot;\theta); \theta \in \Theta\}$. Since this can in principle also be an uncountable infinite set we name our approach *Infinite Kernel Learning* (IKL). The objective we are interested in is the best possible finite MKL solution,

$$\begin{aligned} \inf_{\Theta_f\subset\Theta} \quad \min_{d,v,\xi,b} \quad & \frac{1}{2}\sum_{\theta\in\Theta_f}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i\left(\sum_{\theta\in\Theta_f}\langle v_\theta, \phi_\theta(x_i)\rangle + b\right) \geq 1-\xi_i, \quad i=1,\ldots,n \\ & \xi_i \geq 0, \quad i=1,\ldots,n \\ & \sum_{\theta\in\Theta_f} d_\theta = 1, \qquad d_\theta \geq 0. \end{aligned} \tag{2}$$

The inner minimization problem is the standard finite MKL problem, which is convex. To derive its dual problem we write its Lagrangian

$$\begin{aligned}\mathcal{L}(d,v,b,\xi,\delta,\alpha,\lambda) &= \frac{1}{2}\sum_{\theta\in\Theta_f}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^n \xi_i + \lambda\left(\sum_{\theta\in\Theta_f} d_\theta - 1\right) - \sum_{\theta\in\Theta_f}\delta_\theta d_\theta \\ &\quad - \sum_{i=1}^n \alpha_i\left(y_i\left(\sum_{\theta\in\Theta_f}\langle v_\theta, \phi_\theta(x_i)\rangle + b\right) + \xi_i - 1\right) - \sum_{i=1}^n \eta_i\xi_i,\end{aligned}$$

and take the derivatives w.r.t. to the primal variables

$$\begin{aligned}\tfrac{\partial\mathcal{L}}{\partial d_\theta} &= -\tfrac{1}{2d_\theta^2}\|v_\theta\|^2 + \lambda - \delta_\theta \quad &,\frac{\partial\mathcal{L}}{\partial v_\theta} &= \frac{1}{d_\theta}v_\theta - \sum_{i=1}^n \alpha_i y_i \phi_\theta(x_i) \\ \tfrac{\partial\mathcal{L}}{\partial b} &= -\textstyle\sum_{i=1}^n \alpha_i y_i \qquad &,\frac{\partial\mathcal{L}}{\partial \xi_i} &= C - \alpha_i - \eta_i.\end{aligned}$$

Resubstituting and imposing non-negativity on the Lagrange multipliers corresponding to inequalities, we arrive at the dual problem with the simple form

$$\begin{aligned}\text{(IKL-Dual-1)} \qquad \sup_{\Theta_f\subset\Theta} \quad \max_{\alpha,\lambda} \quad & \sum_{i=1}^n \alpha_i - \lambda \\ \text{subject to} \quad & \alpha\in\mathbb{R}^n,\ \lambda\in\mathbb{R} \\ & 0\le\alpha_i\le C, \quad i=1,\dots,n \\ & T(\theta;\alpha)\le\lambda, \quad \forall\theta\in\Theta_f\end{aligned} \tag{3}$$

where we defined

$$T(\theta;\alpha) = \frac{1}{2}\sum_{i,j=1}^n \alpha_i\alpha_j y_i y_j k(x_i,x_j;\theta). \tag{4}$$

We note that if some point (α^*,λ^*) satisfies the last condition of the program (IKL-Dual-1) for all $\theta\in\Theta$ then it also satisfies the condition for all finite subsets Θ_f thereof. Thus we omit the supremum of (IKL-Dual-1) and extend the program to the following semi-infinite program (SIP)

$$\begin{aligned}\text{(IKL-Dual)} \qquad \max_{\alpha,\lambda} \quad & \sum_{i=1}^n \alpha_i - \lambda \\ \text{subject to} \quad & \alpha\in\mathbb{R}^n,\ \lambda\in\mathbb{R} \\ & 0\le\alpha_i\le C, \quad i=1,\dots,n \\ & T(\theta;\alpha)\le\lambda, \quad \forall\theta\in\Theta.\end{aligned} \tag{5}$$

To show equivalence between both programs it remains to prove that all optimal points of (IKL-Dual) are also optimal for (IKL-Dual-1), in other words that one can construct a finite set Θ_f for the solution of (IKL-Dual) which is also be optimal for (IKL-Dual-1). This is the statement of the following theorem.

Theorem 2.1. ***[(Hettich & Kortanek, 1993)]*** *If for all $\theta\in\Theta$ and for all $\alpha\in[0,C]^n$ we have $T(\theta;\alpha)<\infty$, then there exists a finite set $\Theta_f\subset\Theta$ for which the optimum of (IKL-Dual-1) is taken and the values of (IKL-Dual-1) and (IKL-Dual) are the same.*

Proof. This result is a direct application of Theorem 4.2 in (Hettich & Kortanek, 1993). □

In particular the previous theorem states that there exists an optimal solution of (IKL-Dual) with only a finite number of nonzero d_θ. The final classification function is again of the form

$$f(x) = \text{sign}\left(\sum_{i=1}^n y_i\alpha_i\sum_{\theta\in\Theta_f} d_\theta k(x,x_i;\theta) + b\right). \tag{6}$$

2.1 Constraint Generation for IKL

The dual form (5) of the problem suggests a delayed constraint generation approach to solve it. The set of kernel parameters now defines the set of constraints and in the remainder we will use both terms to refer to Θ. Starting with a finite constraint set $\Theta_0 \subset \Theta$ ones reiterates between the *restricted master problem*, that is the search for optimal α, λ and the *subproblem*, a search for violated constraints indexed by θ. The violated constraints are subsequently included in $\Theta_t \subset \Theta_{t+1} \subset \Theta$. The final algorithm for IKL is summarized in Algorithm 1.

Algorithm 1 Infinite Kernel Learning

Input: Regularization constant C, Kernel parameter set Θ, Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
Output: Parameters α, b, d_θ

1: Select any $\theta_v \in \Theta$ and set $\Theta_0 = \{\theta_v\}$
2: $t \leftarrow 0$
3: **loop**
4: $(\alpha, b, d_\theta, \lambda) \leftarrow$ MKL solution with Θ_t {Solve restricted master problem}
5: $\theta_v \leftarrow \arg\max_{\theta \in \Theta} T(\theta; \alpha)$ {Solve subproblem}
6: **if** $T(\theta_v; \alpha) \leq \lambda$ **then**
7: **break**
8: **end if**
9: $\Theta_{t+1} = \Theta_t \cup \{\theta_v\}$
10: $t \leftarrow t + 1$
11: **end loop**

The restricted master problem in line 4 can be solved using any MKL algorithm like plain gradient descent on the objective function as proposed by (Chapelle et al., 2002; Rakotomamonjy et al., 2007) or the semi infinite linear program (SILP) formulation of (Sonnenburg et al., 2006). Since the parameter λ is the Lagrange multiplier of the equality constraint $\sum_{\theta \in \Theta_t} d_\theta = 1$ it comes as a byproduct of the MKL algorithm.

Efficiently finding violated constraints is essential to the approach. We now state the subproblem explicitly and postpone the detailed discussion of its structure and ways to solve it to Section 3.

Problem 1 (Subproblem)**.** *Given the parameters $0 \leq \alpha_i \leq C$ and training points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, solve*

$$\theta_v = \arg\max_{\theta \in \Theta} T(\theta; \alpha) = \arg\max_{\theta \in \Theta} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta). \tag{7}$$

We can make the following statement, on the convergence of Algorithm 1 which depends on whether the subproblem can be solved.

Theorem 2.2. ***[(Hettich & Kortanek, 1993, Theorem 7.2)]*** *If the subproblem can be solved, Algorithm 1 either stops after a finite number of iterations or has at least one point of accumulation and each one of these points solve (IKL-Dual).*

Proof. Application of (Hettich & Kortanek, 1993, Theorem 7.2). Sketch: compactness (of the admissible set for α) ensures existence of accumulation points. Continuity of the objective and the constraints ensure that these solve (IKL-Dual). □

We want to point out that although Theorem 2.1 guarantees the existence of a finite subset of constraints Θ_f which is sufficient to represent all possible constraints Θ, there is no guarantee that Algorithm 1 will identify this set. But if the algorithm terminates after a finite number of steps (and the subproblem can be solved) it reached *global* optimality. Each strictly violated constraint with $\theta_v \in \Theta$ corresponds to a primal descend direction of the Lagrangian in d_{θ_v}. Therefore, if θ_v is added to the finite set of kernels, the objective function will decrease. To ensure global optimality one therefore needs to ensure that no further descend direction exists which is equivalent to be able to solve the subproblem. All intermediate solutions of the algorithm are primal feasible.

From this viewpoint MKL corresponds to preselecting a set of constraints for the problem and solving the master problem once with this set. The selection task, in other words the solution of Problem 1 is left entirely to the user and the efficiency of MKL depends on how well the user samples from the set of constraints. Importantly any MKL

solution is a subset of IKL, since the set of kernels the user would have chosen can be visited in the subproblem. In IKL the kernel selection criterion is made explicit and optimized over in the subproblem.

2.2 IKL algorithm for finite kernel sets

It is straightforward to see that the IKL algorithm is also applicable to solve the standard MKL problem. In this case at each iteration the subproblem has to be evaluated for each possible kernel choice. Those which violate the constraint $T(\theta;\alpha) \leq \lambda$ are subsequently included in the same manner as it was done for the IKL algorithm. In contrast to solving the MKL problem once using all kernels, for the IKL algorithm not all kernels have to be stored in memory. Furthermore it might be advantageous in terms of speed and stability to optimize the mixing coefficients d_θ on only a small subset of all possible kernel parameters. In the course of the experiments we found that some runs were only tractable using the IKL algorithm in the described way instead of using SimpleMKL (Rakotomamonjy et al., 2007) with all kernels at once. For all experiments we use our own implementation with Coin-IpOpt-3.3.5 (Wächter & Biegler, 2006) and libSVM (Chang & Lin, 2001)).

2.3 Related work

As already mentioned to our knowledge (Argyriou et al., 2006) were the first to note the possibility of an infinite set of base kernels and they also stated the subproblem (Problem 1). We will defer the discussion of the subproblem to the next section and shortly comment on the differences of the Algorithm of (Argyriou et al., 2006) and the IKL Algorithm. We denote with g the objective value of a standard SVM classifier with loss function L

$$g(\alpha, K) = \frac{1}{2}\alpha^T K \alpha + \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{n} \alpha_j k(x_i, x_j)\right). \tag{8}$$

To simplify the notation we also denote with $T(K;\alpha)$ the value of the subproblem for the kernel matrix K and with $K(\theta)$ the kernel matrix for $k(\cdot,\cdot;\theta)$. The algorithm as used in (Argyriou et al., 2006) is summarized in Algorithm 2. At each iteration t it stores the current estimate of the optimal kernel matrix $\hat{K}^t$. The subproblem is solved to search for a new Gram matrix for which the objective will decrease (note that for the special case of only one Gram matrix at the IKL solution it holds $T(K;\alpha) = \lambda$). Subsequently the estimate $\hat{K}^{t+1}$ is computed using a line search between the newly found matrix and the current estimate $\hat{K}^t$. This update step effectively rescales all previously found parameters jointly (Zhang, 2003). In (Argyriou et al., 2006) the line search is implemented using Newtons method which in turn needs a differentiable Loss function like the quadratic loss. Another prominent choice, the hinge loss is not differentiable and therefore one would have to resort to other minimization techniques to solve for the new combination (e.g. any MKL algorithm).

In contrast our algorithm is totally-corrective and iteratively spans a subspace. This means that at each iteration the optimum w.r.t. the entire subspace is found and in particular that each d_θ is updated individually. This allows for multiple pricing, i.e. adding more than one constraints at each iteration of the algorithm.

3 The Subproblem of Infinite Kernel Learning

We now turn our attention to the subproblem of the IKL algorithm. With $T(\theta;\alpha)$ we have an analytic score which guides the selection of kernels. Furthermore we have already seen that being able to solve it is a sufficient condition for global optimality of the solution. The function T consists of two parts. On the one hand there is the outer product of the labels $\mathbf{y}\mathbf{y}^T$ which we will refer to as the *label matrix*. On the other hand the α re-weighted kernel matrix $\alpha\alpha^T * K_\theta$, where the product $*$ denotes the elementwise multiplication. If we use the following inner product between matrices $\langle K_1, K_2\rangle_F = \sum_{i,j=1}^{n} K_1(x_i,x_j)K_2(x_i,x_j)$ we can recognize the subproblem as an alignment problem between matrices, namely as the maximization of $\frac{1}{2}\langle \mathbf{y}\mathbf{y}^T, \alpha\alpha^T * K_\theta\rangle_F$. There is a high alignment if the α reweighted kernel matrix is of a similar structure as the label matrix. The reweighting ensures that the alignment is only measured for support vectors (all non-support vectors have $\alpha_i = 0$) which is reasonable since those are the ones the final classification function is built with.

3.1 Relation to Kernel Target Alignment

Maximizing the alignment between the kernel matrix and the label matrix has been proposed before, usually as a preprocessing step for kernel machines. In (Cristianini, Shawe-Taylor, Elisseeff, & Kandola, 2002) a normalized

Algorithm 2 Algorithm of Argyriou et.al.

Input: Regularization constant C, Kernel parameter set Θ, Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
Output: Parameters α, b, d_θ

Select any $\theta_v \in \Theta$ and set $\Theta_0 = \{\theta_v\}$
$t \leftarrow 0$
$\hat{K}^0 \leftarrow K(\theta_v)$
$(\alpha, b) \leftarrow$ SVM solution with $\hat{K}^0$
loop
  $\theta_v \leftarrow \arg\max_{\theta \in \Theta} T(\theta; \alpha)$ {Solve subproblem}
  if $T(\theta_v; \alpha) \leq T(\hat{K}^t; \alpha)$ **then**
    break
  end if
  Solve $(\alpha, \gamma_v) \leftarrow \arg\max_{\alpha, \gamma} g(\alpha, \gamma K(\theta_v) + (1-\gamma)\hat{K}^t)$
  $\hat{K}^{t+1} \leftarrow \gamma_v K(\theta_v) + (1-\gamma_v)\hat{K}^t$
  $t \leftarrow t + 1$
end loop

version of the above alignment was introduced as *Kernel Target Alignment* (KTA). In our notation it reads

$$\text{(KTA)} \quad A_{\text{KTA}}(K_\theta, \mathbf{y}\mathbf{y}^T) = \frac{\langle K_\theta, \mathbf{y}\mathbf{y}^T \rangle_F}{m\sqrt{\langle K_\theta, K_\theta \rangle_F}}. \tag{9}$$

In (Cristianini et al., 2002) a concentration inequality for this sample-based alignment is presented stating that it is not too dependent on the training set S. Furthermore it is possible to upper bound the generalization error of a Parzen window classifier in terms of KTA. It is concluded that for high KTA one may expect good generalizations and some empirical evidence for this is presented.

Other work on kernel alignments include (Baram, 2005) where *Kernel Polarization* (KP) is proposed which simply omits the normalization term of KTA and thus obtains $A_{\text{KP}}(S, K_\theta, \mathbf{y}\mathbf{y}^T) = m^{-2}\langle K_\theta, \mathbf{y}\mathbf{y}^T \rangle_F$. This is equivalent to the subproblem for constant $\alpha = 1/m$.

An unnormalized version of the alignment was also used in (Crammer, Keshet, & Singer, 2002) to adapt a kernel matrix by explicitly maximizing the alignment in a boosting scheme. The kernels are chosen using the inner product as above, the $\alpha_i \alpha_j$ terms turn into a weight corresponding to the difficulty predicting whether the points x_i and x_j have the same label or not. They restrict the kernel set to outer products and show that solving the weighted alignment turns into a generalized eigenvector problem. It should be possible to adapt this class of kernels for the IKL algorithm.

3.2 Solving the Subproblem

In general, the IKL algorithm is suited for any class of parameterized kernels. The only ingredient is the ability to solve the subproblem 1. The global optimum is only achieved if all violating constraints are identified.

We give up on global optimality for the benefit of being able to use a large class of kernels. Only kernel functions which are differentiable w.r.t. θ are considered. We use the interior point method of (Wächter & Biegler, 2006) to search for local maxima of $T(\cdot; \alpha)$. The search is initialized in various points of the parameter space, including all previously found constraints as well. This worked very well in practice and is used for all the experiments reported in Section 4.

The authors of (Argyriou et al., 2006) proposed the following variant to solve the subproblem for the special case of Gaussian kernels. They note that the subproblem decomposes naturally into a DC problem, that is it can be written as the sum of a convex and a concave function (Horst & Thoai, 1999). Although all C^2-functions can be decomposed into a DC program finding such a decomposition is in general not trivial. The subproblem can be decomposed into

$$T(\theta; \alpha) = \frac{1}{2}\left(\sum_{y_i = y_j} \alpha_i \alpha_j k(x_i, x_j; \theta) - \sum_{y_i \neq y_j} \alpha_i \alpha_j k(x_i, x_j; \theta) \right). \tag{10}$$

For the case of kernels in the form $\exp(-t)$ minimizing this equation corresponds to a minimization of sums of exponentials. In (Argyriou et al., 2006) a cutting plane algorithm is devised to solve this problem. This algorithm finds the optimal solution to the subproblem but is only applicable for the case of few parameters and results are only reported for up to two parameters.

Of course all kinds of optimization techniques can be used to solve for (local) optima of $T(\cdot;\alpha)$, examples are homotopy method like the continuation approach (Rose, 1998). As a remark we want to note that for Lipschitz-continuous kernels, those which satisfy $|k(x,y;\theta_1) - k(x,y;\theta_2)| \leq L|\theta_1 - \theta_2|$ for some constant $L > 0$, one can resort to Lipschitz optimization methods (Horst & Tuy, 1996). This would guarantee global optimality but current methods can deal with only up to 20 variables which in our case correspond to kernel parameters.

An interesting case are kernels which are non-differentiable w.r.t. their parameters or where parameters come from a combinatorial set. For such cases efficient enumeration techniques have to be invented.

4 Experiments

In this section we will present experimental results on a number of benchmark datasets. Before we do so we explain the experimental setup and present an toy example which illustrates the IKL algorithm.

4.1 Algorithmic settings

In practice we make several modifications to Algorithm 1. At each iteration the subproblem returns a set of violating constraints to speed up convergence (we used up to 5 in the experiments). Coefficients d_θ which became zero are pruned out to keep the restricted master problem simpler. To speed up the master and subproblem calls they are warm-started at the previously found solutions. We monitor the change of the objective function and stop the algorithm once the improvement falls below a certain threshold. In the experiments we stopped whenever the decrease is lower than 0.01% of the current objective function value. The implementation is a combination of the interior point solver Coin-IpOpt-3.3.5 (Wächter & Biegler, 2006) and libSVM (Chang & Lin, 2001). The Coin-IpOpt solver is used for both the restricted master problem (standard MKL solution) and as the solver for the subproblem. The MKL program we implemented is SimpleMKL (Rakotomamonjy et al., 2007).

For all runs with a finite number of kernels we either used SimpleMKL or the IKL algorithm since they achieve the same solution. The latter has proven to be faster in the case of many proposal-kernels and high values of the regularization constant C. A detailed comparison of the different solvers in terms of efficiency is subject to future work.

4.2 Considered Kernel Classes

For the experiments we use three different kernel classes which are variants of the well known Gaussian kernel. The first class, in following denoted by **(single)** is the simple Gaussian kernel with isotropic covariance matrix

$$k(x,x';\theta) = \exp\left(-\theta^2 \sum_{d=1}^{D} ([x]_d - [x']_d)^2\right), \tag{11}$$

where $[x]_d$ denotes the d'th element of the vector x. The second kernel class **(separate)** consists of (single) plus all those of the form

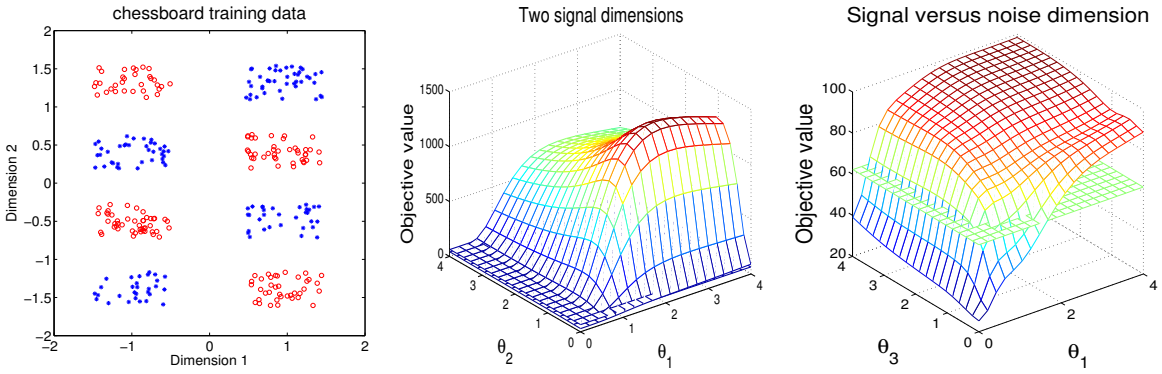
$$k(x,x';\theta_d) = \exp\left(-\theta_d^2([x]_d - [x']_d)^2\right). \tag{12}$$

This kernel measures the distance of two data points x, x' only along the d'th dimension. The final and most general kernel class **(products)** are all Gaussian kernels with an individual parameter θ_d for each dimension d,

$$k(x,x';\theta) = \exp\left(-\sum_{d=1}^{D} \theta_d^2([x]_d - [x']_d)^2\right). \tag{13}$$

It is only with the last kernel that all possible subsets of dimensions can be modelled jointly. Note that this class is too rich to use cross-validation. Just for testing two different choices in each dimension one would need to check 2^D different parameters.

Many other interesting kernels can be considered. We just mention the scaled polynomial $k(x,y;\{\gamma_d\},p) = (1 + \sum_d (x(d)y(d))/\gamma_d^2)^p$ or more general a kernel with rescaling of the input variables $k(x,y;\theta) = k(\theta^T x, \theta^T y)$, see also (Chapelle et al., 2002).



(a) (b) (c)

Figure 1: Chessboard Toy example, (a) distribution of training data in the first two dimensions, (b) IKL subproblem in the first iteration for the two signal dimensions, (c) as (b) for one signal θ_1 and one noise θ_3 dimension.

4.3 Illustrative Toy example

To illustrate the subproblem in conjunction with the product kernels of the previous sections we constructed the following toy example. A total of 300 training points are sampled from a checkerboard pattern as shown in Figure 1(a). Eighteen noise dimensions are added to form a 20 dimensional feature vector. The data is normalized to zero mean and unit variance. Intuitively the best possible kernel would be of the class (products), setting $\theta_d^2 = 0, d = 3, \ldots, 20$ and choosing θ_1 and θ_2 to reflect the distance between the data points in the first two dimensions.

Now the IKL algorithm is started using some initial kernel which is not the "correct" one. In Figure 1(b)+(c) slices of the subproblem function T are shown. In (b) the scaling parameters of the two signal dimensions θ_1 and θ_2 are plotted against the objective value and in (c) θ_1 is plotted against a scaling factor of a noise dimension θ_3. In both cases we have fixed all other scaling factors to zero. The hyperplane associated with the Lagrange multiplier $\lambda \approx 62$ is also shown. All parameters with function values above this hyperplane are violating constraints and thus their inclusion will decrease the objective. The plot of Figure1(b) includes the "true" set of parameters (the ones described above) for the problem and indeed the objective takes a maximum at this point. Note that in Figure1(c) the scale is different, the subproblem function T takes much lower values and thus the expected decrease for each of the shown constraints is much lower. It can also be seen in (c) that the objective function is much more dependent on θ_1 than it is on the scaling factor θ_3, which corresponds to a noise dimension.

In the course of the algorithm, IKL identifies the signal dimensions automatically and selects only a single kernel which achieves a perfect accuracy on a held out test set. To achieve the same result with an SVM or MKL learner one would have to cross-validate over different scaling factors for each dimension or guess the correct scalings for the proposal kernels.

4.4 Benchmark datasets

Up to now and to the best of our knowledge there is no extensive comparison between properly tuned SVM classifiers and those which linearly combine multiple kernels. In this section we will fill this gap. IKL is the limit of MKL and we need to decide whether the added flexibility increases the performance, leads to overfitting or gives qualitatively the same results.

4.4.1 Experimental Setup

Thirteen different binary datasets with up to 100 independent splits (the datasets "image" and "splice" are originally split only 20 times) and the very same experimental setup from (Rätsch, 2001) are used. With five fold CV on the first 5 splits the parameters are determined (the regularizer C and the kernel k for SVM and C for MKL/IKL) which are subsequently used to obtain the results on all splits. The following kernel parameters are considered for all thirteen datasets $\theta^{-1} \in \{1, 2, 3, 5, 10, 20, 30, 40, 50, 75, 100, 125, 150\}$. The regularization constant is selected from the values $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{3}\}$. Five more multiclass datasets were taken from (Duan & Keerthi, 2005) with 20 predefined splits. On each split we use five fold CV on the training set to determine the parameters and then test on the test set (one-versus-rest). Tested kernel parameters are $\theta^{-1} \in$

$\{0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20\}$, the regularization constant is chosen among $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{4}\}$. All data was scaled to have zero mean and unit variance.

The amount of free parameters was varied using the three different classes as described in Section 4.2. For (single) we compared all three methods: SVM, MKL and IKL. For MKL we constructed as proposal kernels all kernel matrices with the kernel parameter in the range as described above. For IKL we allowed for all kernels with $\theta \in [0, 30]$. We found that restricting the admissible range to the one used for MKL and SVM (e.g. $\theta \in [1/150, 1]$) does not alter the results. For (separate) all (single) kernels are used and additionally we applied the same range to every dimension separately. This yields to $13(D + 1)$ proposal kernels for a D dimensional dataset. Finally for (products) we allow all kernels with paramters in $\theta_d \in [0, 30]$. This renders the subproblem to be D dimensional.

4.4.2 Results

The results are shown in Table 1 and Table 2. The four missing values correspond to settings which were too expensive to compute. Even for the results reported here several millions of SVM trainings were performed. We can draw several conclusions from the results. Comparing only the three (single) results we see that the SVM almost always yields to slightly better results than MKL and IKL, which do not differ much. The added flexibility of MKL and IKL to combine more than one kernel seems to be of little use with the possible exception of ABE. In this setup the parameter space seems to be sampled densely enough such that the best kernel parameter for the SVM is well enough approximated. The fact that the SVM yields slightly better or same results seems to indicate that the cross-validation estimate is a more powerful estimator to select kernel parameter than simply minimizing the MKL objective.

Adding the flexibility to model each dimension separately (separate) yields better results only on the datasets Splice and SEG. The most general setting (products) reveals some impressive gains in performance for Image, Splice and SEG, even improving over the MKL-(separate) results. In these cases the possibility to model correlations between different dimensions explicitly yields more discriminative kernels.

For the practitioner there are thus two methods to choose from: SVM because it is much faster than the other two methods and with good performance or IKL because the enlarged kernel class might lead to a significant performance increase.

For some datasets we do observe worse results which are probably due to overfitting behavior (Twonorm, Heart, WAV).

			(single)					**(separate)**		**(products)**	
			SVM	MKL		IKL		MKL		IKL	
Dataset	#dim	#tr / #te	err	err	#k	err	#k	err	#k	err	#k
Banana	2	400/4900	10.5 ± 0.5	10.5 ± 0.5	1.0	10.6 ± 0.5	2.3	10.5 ± 0.5	1.0	10.7 ± 0.5	3.7
Breast-cancer	9	200/77	25.9 ± 4.3	27.9 ± 4.0	2.3	26.9 ± 4.7	2.9	26.7 ± 4.2	4.5	25.7 ± 4.1	16.1
Diabetis	8	468/300	23.2 ± 1.6	24.2 ± 1.9	2.8	23.8 ± 1.7	3.4	24.5 ± 1.6	4.0	24.3 ± 1.8	22.3
Flare-Solar	9	666/400	32.4 ± 1.7	35.1 ± 1.7	1.9	35.0 ± 1.8	2.2	34.3 ± 2.1	2.9	32.8 ± 1.9	2.6
German	20	700/300	23.7 ± 2.1	25.3 ± 2.3	2.0	25.3 ± 2.5	3.4	25.1 ± 2.2	8.3	24.6 ± 2.4	46.1
Heart	13	170/100	15.2 ± 3.1	16.4 ± 3.3	1.0	16.9 ± 3.2	2.5	16.7 ± 4.1	9.0	*20.1 ± 3.6*	28.2
Image	18	130/1010	3.0 ± 0.6	3.3 ± 0.7	1.0	3.4 ± 0.6	5.3	3.0 ± 0.6	1.6	**1.4 ± 0.3**	27.1
Ringnorm	20	400/7000	1.6 ± 0.1	1.6 ± 0.1	1.0	1.6 ± 0.1	1.2	1.7 ± 0.1	2.6	*2.1 ± 0.2*	16.3
Splice	60	1000/2175	10.6 ± 0.7	11.1 ± 0.7	2.0	12.6 ± 0.9	2.0	**6.0 ± 0.4**	24.1	**3.1 ± 0.3**	72.8
Thyroid	5	140/75	4.0 ± 2.2	4.7 ± 2.1	1.0	3.6 ± 2.1	3.2	4.7 ± 2.1	1.0	4.1 ± 2.0	12.7
Titanic	3	150/2051	22.9 ± 1.2	22.4 ± 1.0	1.1	22.5 ± 1.1	2.2	22.4 ± 1.0	1.9	22.4 ± 1.1	5.2
Twonorm	20	400/7000	2.5 ± 0.1	2.5 ± 0.1	2.0	2.6 ± 0.2	2.0	2.5 ± 0.1	3.8	*3.8 ± 0.4*	36.2
Waveform	21	400/4600	10.1 ± 0.5	9.9 ± 0.4	2.9	9.9 ± 0.4	2.5	10.2 ± 0.4	9.7	11.4 ± 0.6	33.7

Table 1: Test error and number of selected kernels on several two class datasets averaged over 100 (20) runs. In bold face are those results with *much* better results than plain SVM and in italic those which are *much* worse. #tr denotes the number of training points, #te the number of test points and #k the average number of selected kernels.

5 Discussion

We generalized MKL to its infinite limit and presented a new algorithm to solve it. Since MKL is a special case of IKL this algorithm can also be used to solve MKL problems. We found it to be more efficient than existing methods like (Rakotomamonjy et al., 2007; Sonnenburg et al., 2006) in the case of many possible kernels (details not reported here). We identified the subproblem as the key ingredient of the kernel learning algorithm. Different to (Argyriou et al., 2006) we solve it in a way which does not guarantee global optimality of the solution. On

				(single)					(separate)		(products)	
				SVM	MKL		IKL		MKL		IKL	
Dataset	#dim	#tr / #te	#cl	err	err	#k	err	#k	err	#k	err	#k
WAV	21	300/4700	3	15.6 ± 1.2	15.5 ± 0.6	2.7	15.8 ± 0.7	2.1	16.4 ± 1.7	13.6	*18.0 ± 1.0*	35.1
SEG	17	500/1810	7	6.5 ± 1.0	6.8 ± 0.9	2.8	6.9 ± 0.9	3.7	$\mathbf{5.0 \pm 0.7}$	8.4	$\mathbf{3.0 \pm 0.5}$	18.0
ABE	16	560/1763	3	1.1 ± 0.3	0.8 ± 0.3	2.5	0.8 ± 0.3	3.0	0.7 ± 0.3	11.3	0.7 ± 0.2	33.8
SAT	36	1500/4935	6	10.4 ± 0.4	10.2 ± 0.3	3.6	10.1 ± 0.4	4.0	n/a		n/a	
DNA	181	500/2686	3	7.7 ± 0.7	7.8 ± 0.7	1.4	7.7 ± 0.8	2.0	n/a		n/a	

Table 2: Test error and number of selected kernels on several multi-class datasets averaged over 20 runs. #cl denotes the number of different classes of the dataset. See also Table 1.

the other hand our approach is much faster and thus applicable to kernel classes with many kernel parameters to choose. As we find in the experiments using very flexible kernel classes seems to be crucial if performance gains are to be expected.

Our experiments are the first large scale comparison between SVM and MKL learning and indicate that for most datasets there is little benefit of linearly combining kernels. Significant improvements over the standard SVM classifier are found for only few datasets. Many results of MKL and IKL are worse than the ones obtained with a standard SVM classifier suggesting an inadequacy of the objective function. A more detailed investigation is subject to future work.

The subproblem provides a handle on how to select new kernels. This opens up the possibility to design problem specific kernels, for example by turning previously separate preprocessing steps into the kernel itself.

Acknowledgments

We are grateful for the discussions with Cheng Soon Ong, Gunnar Rätsch and especially Christoph Lampert who helped to significantly improve the presentation. Both authors were supported by the research project CLASS (IST project 027978).

References

Argyriou, A., Hauser, R., Micchelli, C. A., & Pontil, M. (2006). A DC-programming algorithm for kernel selection. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29*, pp. 41–48.

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, p. 6 New York, NY, USA. ACM.

Baram, Y. (2005). Learning by Kernel Polarization. *Neural Comput.*, **17**(6), 1264–1275.

Bosch, A., Zisserman, A., & Munoz, X. (2007). Representing shape with a spatial pyramid kernel. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 401–408.

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

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing Multiple Parameters for Support Vector Machines. *Mach. Learn.*, **46**(1-3), 131–159.

Crammer, K., Keshet, J., & Singer, Y. (2002). Kernel Design Using Boosting. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15*, pp. 537–544. MIT Press.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. (2002). On Kernel-Target Alignment. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14* Cambridge, MA. MIT Press.

Duan, K., & Keerthi, S. (2005). Which Is the Best Multiclass SVM Method? An Empirical Study. In *Multiple Classifier Systems, 6th International Workshop,MCS 2005, Seaside, CA, USA, June 13-15, 2005, Proceedings*, Vol. 3541 of *Lecture Notes in Computer Science*, pp. 278–285.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 12*, p. 178 Washington, DC, USA. IEEE Computer Society.

Griffin, G., Holub, A., & Perona, P. (2007). Caltech-256 Object Category Dataset. Tech. rep. 7694, California Institute of Technology.

Hettich, R., & Kortanek, K. O. (1993). Semi-infinite programming: theory, methods, and applications. *SIAM Rev.*, **35**(3), 380–429.

Horst, R., & Thoai, N. V. (1999). DC programming: overview. *Journal of Optimization Theory and Applications*, **103**(1), 1–43.

Horst, R., & Tuy, H. (1996). *Global Optimization* (3rd edition). Berlin: Springer-Verlag.

Kimeldorf, G. S., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, **33**, 82–95.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research*, **5**, 27–72.

Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. (2007). More efficiency in multiple kernel learning. In *ICML '07,: Proceedings of the 24th international conference on Machine learning*, pp. 775–782 New York, NY, USA. ACM.

Rätsch, G. (2001). *Robust Boosting via Convex Optimization: Theory and Applications*. Ph.D. thesis, University of Potsdam.

Rose, K. (1998). Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of IEEE*, Vol. 86, pp. 2210–2239.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A Generalized Representer Theorem. In *COLT'01*, Vol. 2111 of *Lecture Notes in Artificial Intelligence*, pp. 416–426. Springer.

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, **7**, 1531–1565.

Wächter, A., & Biegler, L. T. (2006). On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, **106**(1), 25–57.

Zhang, T. (2003). Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, **49**(3), 682–691.

Zien, A., & Ong, C. S. (2007). Multiclass Multiple Kernel Learning. In *24th International Conference on Machine Learning*, pp. 1191–1198 New York, NY, USA. ACM Press.