Infinite Kernel Learning

Peter V. Gehler and Sebastian Nowozin
Max Planck Institute for Biological Cybernetics
72076 Tübingen, Germany
{pgehler,nowozin}@tuebingen.mpg.de

Abstract

In this paper we build upon the Multiple Kernel Learning (MKL) framework and in particular on [2] which generalized it to infinitely many kernels . We rewrite the problem in the standard MKL formulation which leads to a Semi-Infinite Program. We 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 [8]. Furthermore 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 using 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 massive increases in accuracy over SVM/MKL due to the possibility of using a largely increased kernel set. For those cases parameter selection through Cross-Validation or MKL is not applicable.

1 Introduction

In this paper 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 feature vectors $x_i \in \mathbb{R}^D$ and labels $y_i \in \{-1, +1\}$. There are two ingredients which have to be specified prior to SVM training: the kernel function k and the strength of regularization. We will use the notation $k(\cdot, \cdot; \theta)$ to express the dependency of the kernel on some parameters θ. We will think of θ as specifying the *type* of kernel and its corresponding parameters, e.g. RBF and the bandwidth or polynomial and the degree. The amount of regularization is given 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. Usually C and k are chosen according to a Cross Validation estimate.

Multiple Kernel Learning (MKL) [3, 7, 9] is a different approach to select a set of kernel parameters. Here only the parameter C and a set of K so-called *base-kernels* $\{k(\cdot, \cdot; \theta_k)\}_{k=1,\ldots,K}$ have to be specified. The MKL objective function ensures that the final kernel is a convex combination of the proposal kernels $k(x, y; \theta) = \sum_{k=1}^{K} d_k k(x, y; \theta_k)$. The parameters d_k are found simultaneously with the SVM parameters during the minimization of the SVM objective, while the parameters θ_k are kept fixed at all times. It is possible to state this problem as a jointly-convex optimization problem [11, 9]. As already shown by [2] keeping the number of base kernels K fixed is an unnecessary restriction and one can instead search over a possibly infinite set of base-kernels, e.g. all RBF kernels with separate bandwidths $\sigma_i \in [\sigma_{\min}, \sigma_{\max}]$ for each input dimension. Mixture of kernels of different types e.g. polynomial and Gaussians can be encoded in the same way.

2 Infinite Kernel Learning as the optimal Multiple Kernel Learning

We adopt the formulation of the primal objective function for MKL from [11]. However we will write the problem as one optimizing over *general* sets of kernel parameters Θ instead of only finite

sets as is was done in all MKL approaches besides [2]. Since Θ this can in principle also be an uncountable infinite set we dubbed our approach *Infinite Kernel Learning* (IKL). We will use the notation Θ_f for finite sets and Θ for general sets to clarify the difference.

We are interested in is the best possible finite MKL solution.

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

The term inside the $\inf$ term is the standard primal objective function from e.g. [11]. The final classification function is thus of the form $f(x) = \text{sign}(\sum_{i=1}^m y_i \alpha_i \sum_{\theta \in \Theta_f} d_\theta k(x, x_i; \theta) + b)$. For brevity we skip some derivation details[1] and directly write down the corresponding dual and also extend the kernel parameters to the full set Θ and not only finite subsets Θ_f thereof.

$$\begin{aligned} \text{(IKL-Dual)} \quad \max_{\alpha, \lambda} \quad & \sum_{i=1}^n \alpha_i - \lambda \\ \text{sb.t.} \quad & \lambda \in \mathbb{R}, \quad \alpha_i \in [0, C] \quad i = 1, \ldots, n \\ & T(\theta; \alpha) \leq \lambda, \quad \forall \theta \in \Theta, \end{aligned} \tag{2}$$

where we defined

$$T(\theta; \alpha) = \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta). \tag{3}$$

This semi-infinite program (SIP) has as many constraints as there are possible kernel parameters, in the limit infinitely many. The following theorem from the SIP literature ensures that the solution consist of only finitely many kernels with nonzero d_θ.

Theorem 2.1. ***[Hettich & Kortanek [6]]*** *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) restricted to this finite set is achieved. The value of (IKL-Dual) is the same as the one obtained by restricting to Θ_f.*

A new algorithm to solve IKL: Column Generation The dual (2) suggests a delayed constraint generation approach to solve it. Since the kernel parameters become constraints in the dual we use both terms interchangeably. Starting with a few kernel parameters $\Theta_0 \subset \Theta$ one reiterates between the *restricted master problem* and the search for violated constraints indexed by θ which are subsequently included in $\Theta_t \subset \Theta_{t+1} \subset \Theta$. Algorithm 1 summarizes the procedure. Let us shortly explain the ingredients. The restricted master problem is the standard MKL formulation with only finitely many kernel parameters. Thus any MKL algorithm can be used to solve it and we chose to use SimpleMKL [8][2]. The parameter λ is the Lagrange multiplier of the equality constraint $\sum_{\theta \in \Theta_t} d_\theta = 1$ and comes as a byproduct of the MKL algorithm. Efficiently finding violated constraints is essential to the approach and we now state the subproblem explicitly which is a weighted variant of kernel target alignment [4].

Problem 1 (Subproblem)**.** *Given $0 \leq \alpha_i \leq C$ and points $\{x_i, y_i\}$, $i = 1, \ldots, n$, solve*

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

The following theorem gives a convergence guarantee for the case where we can solve the subproblem. Even if the subproblem is not solvable all intermediate solutions of the algorithm are primal feasible.

Theorem 2.2. *[6, 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).*

[1] See the additional material for the complete derivation

[2] We reimplemented it using a mixture of LIBSVM and Coin-IpOpt-3.3.5

Algorithm 1 Infinite Kernel Learning

Require: Regularization constant C, Kernel parameter set Θ, Training set S
Ensure: 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 main difference of Algorithm 1 to the one proposed in [2] is that it is totally-corrective and iteratively spans a subspace. At each iteration the optimum w.r.t. the entire subspace is found. In contrast, the algorithm of [2] performs steps only in the newly found direction [10]. IKL allows for multiple pricing, adding several constraints to the problem at each iterations which speeds up convergence. Both algorithms however solve the same problem.

Solving the Subproblem The IKL algorithm is suited for any class of parametrized kernel families but the most important ingredient is a solver for the resulting subproblem. Consider the finite MKL case. The subproblem can easily be solved since we just need to check finitely many constraints. Therefore Algorithm 1 is also a new MKL algorithm in a straightforward manner. For the case of Gaussian kernels [2] devised a branch-and-bound algorithm to solve it. Whilst this guarantees the global optimum of the subproblem it is feasible only for low dimensional problems and results in [2] are reported for up to two parameters only.

We chose to give up on global optimality for the benefit of a much larger kernel class (see next section). This means that in step 5 in the algorithm we do not search for the *global* optimum θ but search for violating ones, those for which $T(\theta; \alpha) > \lambda$. We employ Newton optimization initialized in all previously found constraints and some heuristic choices for the kernel parameters. This worked very well in practice and is used for all the experiments reported in the experimental section.

3 Experiments

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.

Thirteen different binary datasets with up to 100 independent splits and the experimental setup from [1] are used. Five fold CV on the first 5 splits are used to determine the parameters (C, k for SVM and C for MKL/IKL) which are subsequently used to obtain the results on all splits [3]. Five more multiclass datasets were taken from [5] and split 20 times each. 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).[4] All data was scaled to have zero mean and unit variance. Gaussian kernels of the following form are used

$$k(x, x'; \{\gamma_1, \ldots, \gamma_D\}) = \exp\left(-\sum_{d=1}^{D} \gamma_d [x]_d [x']_d\right), \ \gamma_d \in [\gamma_{\min}, \gamma_{\max}] \subset \mathbb{R}_+, \tag{5}$$

where $[x]_d$ denotes the dth element of x. We varied the free parameters: **(single)** isotropic Gaussian (all γ_d equal) **(separate)** as (single) + each dimension separately (e.g. $\gamma = (0, 0, \gamma_3, 0, \ldots, 0)$) and **(products)** all possible kernels with parameters in the interval $\gamma_d \in [0, 30]$. With the latter choice it is possible to rescale the data in *every* dimension separately and model dependencies between subsets of dimensions ignoring others altogether ($\gamma_d = 0$). This renders the subproblem (3) to be D dimensional. Note that there are far too many parameters to perform CV or preselect choices for MKL. For **SVM** we used a grid of kernel parameters (see footnotes), **MKL** had access to all those kernels simultaneously, thus 13 for the binary datasets for (single) and 13*(D+1) for (separate).

[3] 13 kernel parameters are tested $1/([1, 2, 3, 5, 10, 20, 30, 40, 50, 75, 100, 125, 150]^2$

[4] 10 kernel parameters are tested $1/[0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20]^2$.

The results are shown in Table 1. Missing values correspond to settings which were far too expensive to compute, even for the results reported here several millions of SVM trainings were performed. We draw several conclusions from the results. Comparing only the (single) results we see that the SVM almost always yields to slightly better results than MKL/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 setting the parameter space seems to be sampled densely enough such that the results are very close. Adding the flexibility to model each dimension separately (separate) yields better results only on Splice and SEG. The most general setting (products) reveals some immense gains in performance, namely for Image, Splice and SEG. 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). During the course of the experiments

				(single)					**(separate)**		**(products)**	
				SVM	MKL		IKL		MKL		IKL	
Dataset	#dim	#tr / #te	#cl	err	err	#k	err	#k	err	#k	err	#k
Banana	2	400/4900	2	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	2	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	2	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	2	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	2	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	2	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	2	3.0 ± 0.6	3.3 ± 0.7	1.0	3.4 ± 0.6	5.3	3.0 ± 0.6	1.6	$\mathbf{1.4 \pm 0.3}$	27.1
Ringnorm	20	400/7000	2	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	2	10.6 ± 0.7	11.1 ± 0.7	2.0	12.6 ± 0.9	2.0	$\mathbf{6.0 \pm 0.4}$	24.1	$\mathbf{3.1 \pm 0.3}$	72.8
Thyroid	5	140/75	2	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	2	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	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	2	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
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 1: Test error and number of selected kernels on several 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.

we found that some runs were only tractable using the IKL algorithm: starting with one kernel and searching for new ones to include instead of SimpleMKL [8] with all kernels at once. This occurred especially for high values of C and cases where Gram matrices are very similar. The IKL algorithm is also faster in the case of many base kernels.

4 Conclusion

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 [8, 9] in the case of many possible kernels (details not reported here). Our experiments are the first large scale comparison between SVM and MKL learning and indicate there there is little benefit of linearly combining kernels. With the largely increased flexibility of IKL to search over general kernel classes not available to MKL/SVM we reported significant performance gains on some datasets. Therefore it seems crucial to use such general kernel classes if performance gains are to be expected. The subproblem provides a handle on how to select new kernels. This opens up the possibility to design problem specific kernels, e.g. turning preprocessing steps to kernel parameters.

References

[1] `http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm`.

[2] Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Pontil. A dc-programming algorithm for kernel selection. In *ICML '06*, 2006.

[3] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML '04*, 2004.

[4] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *NIPS' 01*, 2002.

[5] K. Duan and S.S. Keerthi. Which is the best multiclass svm method? an empirical study. In *Multiple Classifier Systems*, volume 3541 of *Lecture Notes in Computer Science*, pages 278–285, 2005.

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

[7] 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.

[8] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *ICML '07*, 2007.

[9] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 2006.

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

[11] A. Zien and C.S. Ong. Multiclass multiple kernel learning. In *ICML*. ACM Press, 06 2007.