Easy multiple kernel learning

Fabio Aiolli and Michele Donini

University of Padova - Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy

Abstract. The goal of Multiple Kernel Learning (MKL) is to combine kernels derived from multiple sources in a data-driven way with the aim to enhance the accuracy of a kernel based machine. In this paper, we propose a time and space efficient MKL algorithm that can easily cope with hundreds of thousands of kernels and more. We compared our algorithm with other baselines plus three state-of-the-art MKL methods showing that our approach is often superior.

1 Introduction

Multiple Kernel Learning (MKL) is a general paradigm used to learn kernels. The kernel computed are (generally linear) combinations of previously defined *weak* kernels trained using available data. See for example [1] for a quite exhaustive survey. In this paper, we focus on MKL with positive and linear combination parameters, that is, in the form $\mathbf{K} = \sum_{r=1}^{R} \eta_r \mathbf{K}_r, \ \ \eta_r \geq 0$. According to [1], the majority of existing MKL approaches can be divided into the following two categories. *Fixed or heuristic rule* based techniques typically result scalable with respect to the number of kernels combined but their effectiveness will critically depend on the domain at hand. On the other side, *optimization based* approaches learn the combination parameters by solving an optimization problem. Generally speaking, best performing MKL approaches necessitate of complex optimization (e.g. SDP, QCQP) which will soon exhaust memory when copying with hundreds of examples and hundreds of weak kernels. Here, we propose an algorithm able to effectively and efficiently optimize the separation between positive and negative examples. We empirically demonstrate the effectiveness of the method proposed which is almost always more accurate than the baselines.

2 The KOMD Algorithm

We consider a classification problem with training examples $\{(\mathbf{x}_1, y_i), \ldots, (\mathbf{x}_l, y_l)\}$, and test examples $\{(\mathbf{x}_{l+1}, y_{l+1}), \ldots, (\mathbf{x}_L, y_L)\}$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$. We use $\mathbf{X} \in \mathbb{R}^{L \times m}$ to denote the matrix where examples are arranged in rows and $\mathbf{y} \in \mathbb{R}^L$ the vector of labels. The matrix $\mathbf{K} \in \mathbb{R}^{L \times L}$ denotes the complete kernel matrix containing the kernel values of each (training and test) data pair. Further, we indicate with an hat, like for example $\hat{\mathbf{X}} \in \mathbb{R}^{l \times m}$, $\hat{\mathbf{y}} \in \mathbb{R}^l$, and $\hat{\mathbf{K}} \in \mathbb{R}^{l \times l}$, the submatrices (or subvectors) obtained considering training examples only.

Given a training set, we consider the domain $\hat{\Gamma}$ of probability distributions $\gamma \in \mathbb{R}^l_+$ defined over the sets of positive and negative training examples. More formally: $\hat{\Gamma} = \{\gamma \in \mathbb{R}^l_+ \ \ | \sum_{i \in \oplus} \gamma_i = 1, \sum_{i \in \ominus} \gamma_i = 1\}$.

In [2] a game theoretic interpretation of the problem of margin maximization and an algorithm called "Kernel method for the Optimization of the Margin Distribution" (KOMD) have been proposed. The classification task is posed as a two-player zero-sum game and it is shown how this zero-sum game is equivalent to an hard SVM which can be solved efficiently by optimizing a simple linearly constrained convex function on variables $\gamma \in \hat{\Gamma}$, namely,

$$\text{minimize}_{\gamma \in \hat{\Gamma}} D(\gamma) := \gamma^\top \hat{\mathbf{Y}} \hat{\mathbf{K}} \hat{\mathbf{Y}} \gamma.$$

The vector $\gamma^* \in \hat{\Gamma}$ that minimizes $D(\gamma)$ identifies the two nearest points in the convex hulls of positive and negative examples, in the feature space of kernel $\mathbf{K}$. Moreover, a quadratic regularization over γ, namely $R_\gamma(\gamma) = \gamma^\top \gamma$, is introduced, that makes the player to prefer optimal distributions (strategies) with low variance. The final best strategy for γ will be given by solving the optimization problem $\min_{\gamma \in \hat{\Gamma}} (1-\lambda) D(\gamma) + \lambda R_\gamma(\gamma)$. A correct selection of the parameter $\lambda \in [0, 1]$ (usually made by validating on training data) is fundamental. In KOMD, the evaluation on a new generic example $\mathbf{x}$ is obtained by: $f(\mathbf{x}) = \sum_i y_i \gamma_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) = \mathbf{K}_{tr}(\mathbf{x}) \hat{\mathbf{Y}} \gamma$, where $\mathbf{K}_{tr}(\mathbf{x}) = [\mathbf{K}(\mathbf{x}_1, \mathbf{x}), \ldots, \mathbf{K}(\mathbf{x}_l, \mathbf{x})]^\top$.

3 EasyMKL

In MKL we want to find the best combination parameters for a set of predefined kernel matrices. In our context, this is done by learning a coefficient vector η that forms the combined kernel according to: $\mathbf{K} = \sum_{r=1}^{R} \eta_r \mathbf{K}_r, \;\; \eta_r \geq 0$. Clearly, we must restrict the possible choices of such a matrix $\mathbf{K}$ and this can be made by regularizing the learning process. So, we pose the problem of learning the kernel combination as a min-max problem over variables γ and η. Specifically, we propose to maximize the distance between positive and negative examples with a unitary norm vector η as the weak kernel combination vector, that is

$$\max_{||\eta||=1} \min_{\gamma \in \Gamma} (1-\lambda) \gamma^\top \hat{\mathbf{Y}} (\sum_r \eta_r \hat{\mathbf{K}}_r) \hat{\mathbf{Y}} \gamma + \lambda ||\gamma||^2 = \min_{\gamma \in \Gamma} \max_{||\eta||=1} Q(\eta, \gamma),$$

where $Q(\eta, \gamma) = (1-\lambda) \eta^\top \mathbf{d}(\gamma) + \lambda ||\gamma||^2$ and the *r-th* entry of $\mathbf{d}(\gamma)$ is $\mathbf{d}_r(\gamma) = \gamma^\top \hat{\mathbf{Y}} \hat{\mathbf{K}}_r \hat{\mathbf{Y}} \gamma$.

Now, it is possible to see that, the vector η^* maximizing the function $Q(\eta, \gamma)$ above has a simple analytic solution: $\eta^* = \frac{\mathbf{d}(\gamma)}{||\mathbf{d}(\gamma)||}$. Plugging this solution into the min-max problem, we obtain

$$\min_\gamma Q(\eta^*, \gamma) = \min_\gamma (1-\lambda) ||\mathbf{d}(\gamma)|| + \lambda ||\gamma||^2.$$

The optimal γ will be the (regularized) minimizer of the 2-norm of the vector of distances. This minimizer is not difficult to find as this is a convex function though not quadratic. In order to simplify the problem further we prefer to minimize an upper-bound instead corresponding to the 1-norm of the vector of distances:

$$\min_\gamma (1-\lambda) ||\mathbf{d}(\gamma)||_1 + \lambda ||\gamma||^2 = \min_\gamma (1-\lambda) \gamma^\top \hat{\mathbf{Y}} (\sum_r \hat{\mathbf{K}}_r) \hat{\mathbf{Y}} \gamma + \lambda ||\gamma||^2.$$

Interestingly, the minimization problem is the same as the KOMD problem where the kernel matrix has been substituted with the simple sum of the weak kernel matrices. This will turn out to be very important for the efficiency (especially in terms of space) of the method as will be explained later. In the following, we refer to this algorithm as *EasyMKL*. A final consideration we can do here is that the quality of a kernel does not change if it is multiplied by a positive constant. However, this formulation makes particularly clear that, if kernels with different traces are present in the combination, they have unequal impact in the MKL optimization problem. Thus, if not differently motivated, this should be avoided.

4 Experiments and Results

Experiments have been performed comparing the proposed methods with other state-of-the-art MKL algorithms with respect to accuracy and computational performance. A total of 4 datasets and 9 different methods have been tested. Benchmark datasets with different characteristics have been used. In particular, the Splice dataset [3] (60 features, 1,000 examples), the Batch2 dataset [4] (128 features, 1,244 examples), the Mush dataset [3] (112 features, 2,000 examples) and the Gisette dataset [5] (5,000 features, 1,000 examples). All data have been scaled to $[-1, +1]$ interval and the number of training examples selected for the training part is approximately 10% of the dataset. We decided to use relatively small training sets as in this case the final accuracy is less influenced by the classifier (on which the combined kernel is applied) and more on the real quality of a kernel. Further, we preferred having larger test sets and many different splits of a single dataset in order to improve the significance of the overall evaluation.

4.1 RBF based weak kernels

In this paper, we want to demonstrate that we can perform kind of feature selection via MKL by selecting a large set of weak kernels which are individually defined using a small subset of features. Let $d \in \mathbb{N}$ and $\beta > 0$ two parameters, then the r^{th} kernel is constructed by random picking of a bag (replica are allowed) of features F_r, such that $|F_r| \leq d$ and an RBF based weak kernel is defined according to $\mathbf{K}_r(\mathbf{x}_i, \mathbf{x}_j) = \prod_{f \in F_r} e^{-\frac{\beta}{|F_r|}(\mathbf{x}_i^{(f)} - \mathbf{x}_j^{(f)})^2}$.

4.2 Methods

We have then considered three baseline methods for our experiments. The first method is the classical (**SVM**) trained with all the features. This is only reported for the sake of completeness and just to give an idea of the difficulty of the datasets. Note that, the feature space in this case is different from MKL methods and results are not comparable. The second method, called Random Kernel (**Random**), consists of random picking from available kernels, which is equivalent to set only one η_r equal to 1 and all the others equal to 0. We decided to insert this baseline as an indicator of how much information is brought from single kernels. Intentionally, the performance of this baseline could be really poor. Finally, the last baseline method is the Average Kernel (**Average**) where

the same weight η_r is given to all the available weak kernels. Despite its simplicity, it is known (see [6]) that this is a strong baseline that beats other more advanced techniques quite often and can obtain very good results. We have then considered three state-of-the-art algorithms of MKL with SVM. All these three algorithms are in the optimization based family of MKL methods. A MATLAB implementation[1] of these methods by Mehmet Gönen and Ethem Alpaydin [1] has been used.

- **Simple MKL (SMKL)** An iterative algorithm by Rakotomamonjy [7] that implements a linear approach with kernel weights in a simplex.
- **Generalized MKL (GMKL)** The second algorithm, by Varma and Babu [8] exploits a nonlinear approach different from SMKL but, in general, with better results [1].
- **Lasso-based MKL (GLMKL)** This algorithm by Kloft [9] and Xu [10] considers a regularization of kernel weights with 1-norm and finds the kernel weights by solving a small QP problem at each iteration.
- **SMO Projected Gradient Descent Generalized MKL (SPG-GMKL)** This particular algorithm [11] is a state-of-the-art method with respect to the computational performance.

4.3 AUC Experiments and Results

The area the under curve metric (AUC) has been used to evaluate the quality in classification of the different combined kernels. Weak kernels have been generated according to the method depicted in Section 4.1. A number of $10,000$ weak kernels have been constructed for each dataset. Two different values for $d \in \{5, 10\}$ have been used in the generation algorithm with the RBF parameter β_0 obtained by model selection on a standard SVM. The experimental setting is summarized below:

1. Repeat for $r = 1$ to $R = 10,000$: pick a random $p \in \{1, ..., d\}$ and generate a weak kernel K_r using p features picked randomly (with replacement).
2. Combine $\{\mathbf{K}_r\}_{r=1}^{R}$ with a MKL algorithm to obtain a kernel $\mathbf{K}$.
3. Rank test examples using $\mathbf{K}$ and SVM (KOMD for the proposed method) and evaluate the quality of the ranking with the AUC metric.

The same experiments have been repeated 1,000 times averaging the AUC results in order to improve the significance of the evaluation. In order to set our experiments as fair as possible, the KOMD algorithm has been used with the kernel produced by our MKL method while SVM has been used on all the others. However, in [2] it has been shown that these two methods have in fact similar accuracy.

[1] `http://www.cmpe.boun.edu.tr/~gonen/mkl`.

RBF kernels with all the features have been used to train the SVM baseline which obtained an AUC value of 86.12% for the Splice dataset, 95.20% for the Batch2 dataset and 99.52% for the Mush dataset. The results obtained with the MKL algorithms are summarized in Table 1 ($d = 5$ and $d = 10$) which shows that standard MKL methods do not have performances significantly better than the simple kernel averaging. On the other side, EasyMKL have significantly better AUC in two out of three datasets. This trend is confirmed in Table 2 where the proposed method is always significantly better than the average baseline. Unfortunately, given the relatively large number of features of the Gisette dataset, it was not possible to run state-of-the-art MKL methods on this data without running out the memory.

	Average AUC (RBF $d = 5$)			Average AUC (RBF $d = 10$)		
Algorithm	*Splice*	*Batch2*	*Mush*	*Splice*	*Batch2*	*Mush*
Random	52.61%	81.14%	28.58%	56.76%	85.30%	44.58%
Average	86.56%	95.35%	**98.45**%	90.42%	94.70%	**98.86**%
SMKL	83.48%	94.82%	97.49%	86.84%	95.07%	97.48%
GMKL	83.70%	92.53%	97.39%	84.48%	94.92%	96.99%
GLMKL	85.36%	94.54%	96.38%	86.41%	94.18%	95.25%
EasyMKL	**87.87**%	**98.98**%	97.91%	**91.19**%	**97.08**%	98.29%

Table 1: AUC results on three datasets of various MKL methods and baselines (RBF feature subset $d = 5$ and $d = 10$).

Algorithm	RBF $d = 5$	RBF $d = 10$	RBF $d = 20$
Average	95.52%	94.72%	93.65%
EasyMKL	**95.83**%	**95.40**%	**94.64**%

Table 2: Average AUC on the Gisette (NIPS2003 feature selection challenge) of EasyMKL (with RBF subset) and comparison with the Average baseline.

4.4 Performance Experiments and Results

We also performed experiments[2] to evaluate the computational time and the memory used by EasyMKL. We compared our algorithm with a state-of-the-art algorithm called SPG-GMKL [11]. A C++ implementation of SPG-GMKL provided by the authors[3] has been used. SPG-GMKL with this implementation is more than 100 times faster than SimpleMKL [11]. The Splice dataset has been selected for this experiment with 100 training examples. A variable number of RBF Kernels has been generated with a parameter β picked randomly in $]0, 1[$ and using all the 60 features of the Splice dataset. We have studied the performance in time and memory, fixed an upper bound of 2 GB for the memory. Based on our experiments, time complexity of SPG-GMKL is linear with the

[2]For these experiments we used a CPU Intel Core i7-3632QM @ 2.20GHz 2.20GHz
[3]http://www.cs.cornell.edu/∼ashesh/pubs/code/SPG-GMKL/download.html

number of kernels, with a constant of 0.15 seconds per kernel. EasyMKL has a linear increase in time too, with 6.62 seconds per kernel. The memory used by SPG-GMKL has a sub-linear growth with the number of kernels and has reached 2 GB with only 400 kernels. The optimization in our algorithm consists of KOMD optimization on a sum of kernels. Hence there is not need to store all the matrices and it just requires the memory to store a couple of kernel matrices. It follows that, with our method, we can use an *unlimited* number of different kernels with only a small memory requirement (e.g. 47 MB for the Splice dataset). Note that, the efficiency of our solution could be improved with an optimized implementation coded in C++ instead of ours coded in Python.

5 Conclusion

In this paper we proposed the EasyMKL algorithm which is able to cope efficiently with a very large number of different kernels. Experiments have shown that the proposed method is more accurate than other MKL state-of-the-art methods. Concerning time and memory requirements with respect to the number of combined kernels, EasyMKL uses only a fixed amount of memory, hence its complexity is constant, and it has only linear time complexity.

References

[1] Mehmet Gonen and Ethem Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.

[2] Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. A kernel method for the optimization of the margin distribution. In *ICANN (1)*, pages 305–314, 2008.

[3] K. Bache and M. Lichman. Uci machine learning repository, 2013.

[4] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Raman Huerta. Chemical gas sensor drift compensation using classifier ensembles, 2012.

[5] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in NIPS 17*, pages 545–552. MIT Press, Cambridge, MA, 2005.

[6] Xinxing Xu, Ivor W. Tsang, and Dong Xu. Soft margin multiple kernel learning. *IEEE Trans. Neural Netw. Learning Syst.*, 24(5):749–761, 2013.

[7] Alain Rakotomamonjy, Francis R. Bach, St'ephane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 208.

[8] Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors. *Proceedings of ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*. ACM, 2009.

[9] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Non-sparse regularization and efficient training with multiple kernels. *CoRR*, abs/1003.0079, 2010.

[10] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pages 1175–1182, 2010.

[11] A. Jain, S. V. N. Vishwanathan, and M. Varma. Spg-gmkl: Generalized multiple kernel learning with a million kernels. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2012.