Multitask Learning Using Regularized Multiple Kernel Learning

Mehmet Gönen[1], Melih Kandemir[1], and Samuel Kaski[1,2]

[1] Aalto University School of Science
Department of Information and Computer Science
Helsinki Institute for Information Technology HIIT
[2] University of Helsinki
Department of Computer Science
Helsinki Institute for Information Technology HIIT

Abstract. Empirical success of kernel-based learning algorithms is very much dependent on the kernel function used. Instead of using a single fixed kernel function, *multiple kernel learning* (MKL) algorithms learn a combination of different kernel functions in order to obtain a similarity measure that better matches the underlying problem. We study *multitask learning* (MTL) problems and formulate a novel MTL algorithm that trains coupled but nonidentical MKL models across the tasks. The proposed algorithm is especially useful for tasks that have different input and/or output space characteristics and is computationally very efficient. Empirical results on three data sets validate the generalization performance and the efficiency of our approach.

Keywords: kernel machines, multilabel learning, multiple kernel learning, multitask learning, support vector machines.

1 Introduction

Given a sample of N independent and identically distributed training instances $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i$ is a D-dimensional input vector and y_i is its target output, kernel-based learners find a decision function in order to predict the target output of an unseen test instance $\boldsymbol{x}$ [10,11]. For example, the decision function for binary classification problems (i.e., $y_i \in \{-1, +1\}$) can be written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

where the kernel function ($k\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$) calculates a similarity metric between data instances. Selecting the kernel function is the most important issue in the training phase; it is generally handled by choosing the best-performing kernel function among a set of kernel functions on a separate validation set.

In recent years, *multiple kernel learning* (MKL) methods have been proposed [4], for learning a combination k_η of multiple kernels instead of selecting one:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\eta}) = f_\eta(\{k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)_{m=1}^P\}; \boldsymbol{\eta})$$

where the combination function ($f_\eta \colon \mathbb{R}^P \to \mathbb{R}$) forms a single kernel from P base kernels using the parameters $\boldsymbol{\eta}$. Different kernels correspond to different notions of similarity and instead of searching which works best, the MKL method does the picking for us, or may use a combination of kernels. MKL also allows us to combine different representations possibly from different sources or modalities.

When there are multiple related machine learning problems, tasks or data sets, it is reasonable to assume that also the models are related and to learn them jointly. This is referred to as *multitask learning* (MTL). If the input and output domains of the tasks are the same (e.g., when modeling different users of the same system as the tasks), we can train a single learner for all the tasks together. If the input and/or output domains of the tasks are different (e.g., in multilabel classification where each task is defined as predicting one of the labels), we can share the model parameters between the tasks while training.

In this paper, we formulate a novel algorithm for *multitask multiple kernel learning* (MTMKL) that enables us to train a single learner for each task, benefiting from the generalization performance of the overall system. We learn similar kernel functions for all of the tasks using separate but regularized MKL parameters, which corresponds to using a similar distance metric for each task. We show that such coupled training of MKL models across the tasks is better than training MKL models separately on each task, referred to as *single-task multiple kernel learning* (STMKL).

In Section 2, we give an overview of the related work. Section 3 explains the key properties of the proposed algorithm. We then demonstrate the performance of our MTMKL method on three data sets in Section 4. We conclude by a summary of the general aspects of our contribution in Section 5.

We use the following notation throughout the rest of this paper. We use boldface lowercase letters to denote vectors and boldface uppercase letters to denote matrices. The i and j are used as indices for the training instances, r and s for the tasks, and m for the kernels. The T and P are the numbers of the tasks and the kernels to be combined, respectively. The number of training instances in task r is denoted by N^r.

2 Related Work

[2] introduces the idea of multitask learning, in the sense of learning related tasks together by sharing some aspects of the task-specific models between all the tasks. The ultimate target is to improve the performance of each individual task by exploiting the partially related data points of other tasks.

The most frequently used strategy for extending discriminative models to multitask learning is by following the hierarchical Bayes intuition of ensuring similarity in parameters across the tasks by binding the parameters of separate tasks [1]. Parameter binding typically involves a coefficient to tune the similarity between the parameters of different tasks. This idea is introduced to kernel-based algorithms by [3]. In essence, they achieve parameter similarity by decomposing the hyperplane parameters into shared and task-specific components. The model reduces to a single-kernel learner with the following kernel function:

$$\widehat{k}(\boldsymbol{x}_i^r, \boldsymbol{x}_j^s) = (1/\nu + \delta_r^s) k(\boldsymbol{x}_i^r, \boldsymbol{x}_j^s)$$

where ν determines the similarity between the parameters of different tasks and δ_r^s is 1 if $r = s$ and 0 otherwise. The same model can be extended to MKL using a combined kernel function:

$$\widehat{k_\eta}(\boldsymbol{x}_i^r, \boldsymbol{x}_j^s; \boldsymbol{\eta}) = (1/\nu + \delta_r^s) k_\eta(\boldsymbol{x}_i^r, \boldsymbol{x}_j^s; \boldsymbol{\eta}) \tag{1}$$

where we can learn the combination parameters $\boldsymbol{\eta}$ using standard MKL algorithms. This task-dependent kernel approach has three disadvantages: (a) It requires all tasks to be in a common input space to be able to calculate the kernel function between the instances of different tasks. (b) It requires all tasks to have similar target outputs to be able to capture them in a single learner. (c) It requires more time than training separate but small learners for each task.

There are some recent attempts to integrate MTL and MKL in multilabel settings. [5] uses multiple hypergraph kernels with shared parameters across the tasks to learn multiple labels of a given data set together. Learning the large set of kernel parameters in this special case of the multilabel setup requires a computationally intensive learning procedure. In a similar study, [12] suggests decomposing the kernel weights into shared and label-specific components. They develop a computationally feasible, but still intensive, algorithm for this model. In a multitask setting, [9] proposes to use the same kernel weights for each task.

[6] proposes a feature selection method that uses separate hyperplane parameters for the tasks and joins them by regularizing the weights of each feature over the tasks. This method enforces the tasks to use each feature either in all tasks or in none. [7] uses the parameter sharing idea to extend the large margin nearest neighbor classifier to multitask learning by decomposing the covariance matrix of the Mahalanobis metric into task-specific and task-independent parts. They report that using different but similar distance metrics for the tasks increases generalization performance.

Instead of binding different tasks using a common learner as in [3], we propose a general and computationally efficient MTMKL framework that binds the different tasks to each other through the MKL parameters, which is discussed under multilabel learning setup by [12]. They report that using different kernel weights for each label does not help and suggest to use a common set of weights for all labels. We allow the tasks to have their own learners in order to capture the task-specific properties and to use similar kernel functions (i.e., separate but regularized MKL parameters), which corresponds to using similar distance metrics as in [7], in order to capture the task-independent properties.

3 Multitask Learning Using Multiple Kernel Learning

There are two possible approaches to integrate MTL and MKL under a general and computationally efficient framework: (a) using common MKL parameters for each task, and (b) using separate MKL parameters but regularizing them in order to have similar kernel functions for each task. The first approach is also discussed in [9] and we use this approach as a baseline comparison algorithm.

Sharing exactly the same set of kernel combination parameters might be too restrictive for weakly correlated tasks. Instead of using the same kernel function, we can learn different kernel combination parameters for each task and regularize them to obtain similar kernels. Model parameters can be learned jointly by solving the following min-max optimization problem:

$$\underset{\{\boldsymbol{\eta}^r \in \mathcal{E}\}_{r=1}^T}{\text{mininimize}} \; \mathcal{O}_\eta = \left\{ \underset{\{\boldsymbol{\alpha}^r \in \mathcal{A}^r\}_{r=1}^T}{\text{maximize}} \; \Omega(\{\boldsymbol{\eta}^r\}_{r=1}^T) + \sum_{r=1}^T J^r(\boldsymbol{\alpha}^r, \boldsymbol{\eta}^r) \right\} \tag{2}$$

where $\Omega(\cdot)$ is the regularization term calculated on the kernel combination parameters, the $\mathcal{E}$ denotes the domain of the kernel combination parameters, $J^r(\cdot, \cdot)$ is the objective function of the kernel-based learner of task r, which is generally composed of a regularization term and an error term, and the $\mathcal{A}^r$ is the domain of the parameters of the kernel-based learner of task r.

If the tasks are binary classification problems (i.e., $y_i^r \in \{-1, +1\}$) and the squared error loss is used implying least squares support vector machines, the objective function and the domain of the model parameters of task r become

$$J^r(\boldsymbol{\alpha}^r, \boldsymbol{\eta}^r) = \sum_{i=1}^{N^r} \alpha_i^r - \frac{1}{2} \sum_{i=1}^{N^r} \sum_{j=1}^{N^r} \alpha_i^r \alpha_j^r y_i^r y_j^r \left(k_\eta^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r; \boldsymbol{\eta}^r) + \frac{\delta_i^j}{2C} \right)$$

$$\mathcal{A}^r = \left\{ \boldsymbol{\alpha}^r \colon \sum_{i=1}^{N^r} \alpha_i^r y_i^r = 0, \; \alpha_i^r \in \mathbb{R} \;\; \forall i \right\}$$

where C is the regularization parameter. If the tasks are regression problems (i.e., $y_i^r \in \mathbb{R}$) and the squared error loss is used implying kernel ridge regression, the objective function and the domain of the model parameters of task r are

$$J^r(\boldsymbol{\alpha}^r, \boldsymbol{\eta}^r) = \sum_{i=1}^{N^r} \alpha_i^r y_i^r - \frac{1}{2} \sum_{i=1}^{N^r} \sum_{j=1}^{N^r} \alpha_i^r \alpha_j^r \left(k_\eta^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r; \boldsymbol{\eta}^r) + \frac{\delta_i^j}{2C} \right)$$

$$\mathcal{A}^r = \left\{ \boldsymbol{\alpha}^r \colon \sum_{i=1}^{N^r} \alpha_i^r = 0, \; \alpha_i^r \in \mathbb{R} \;\; \forall i \right\}.$$

If we use a convex combination of kernels, the domain of the kernel combination parameters becomes

$$\mathcal{E} = \left\{ \boldsymbol{\eta} \colon \sum_{m=1}^P \eta_m = 1, \; \eta_m \geq 0 \;\; \forall m \right\}$$

and the combined kernel function of task r with the convex combination rule is

$$k_\eta^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r; \boldsymbol{\eta}^r) = \sum_{m=1}^P \eta_m^r k_m^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r).$$

Similarity between the combined kernels is enforced by adding an explicit regularization term to the objective function. We propose the sum of the dot products between kernel combination parameters as the regularization term:

$$\Omega(\{\boldsymbol{\eta}^r\}_{r=1}^T) = -\nu \sum_{r=1}^{T} \sum_{s=1}^{T} \langle \boldsymbol{\eta}^r, \boldsymbol{\eta}^s \rangle. \tag{3}$$

Using a very small ν value corresponds to treating the tasks as unrelated, whereas a very large value enforces the model to use similar kernel combination parameters across the tasks. The regularization function can also be interpreted as the negative of the total correlation between the kernel weights of the tasks and we want to minimize the negative of the total correlation if the tasks are related. Note that the regularization function is concave but efficient optimization is possible thanks to the bounded feasible sets of the kernel weights.

The min-max optimization problem in (2) can be solved using an alternating optimization procedure analogous to many MKL algorithms in the literature [8,13,14]. Algorithm 1 summarizes the training procedure. First, we initialize the kernel combination parameters $\{\boldsymbol{\eta}^r\}_{r=1}^T$ uniformly. Given $\{\boldsymbol{\eta}^r\}_{r=1}^T$, the problem reduces to training T single-task single-kernel learners. After training these learners, we can update $\{\boldsymbol{\eta}^r\}_{r=1}^T$ by performing a projected gradient-descent steps to order to satisfy two constraints on the kernel weights: (a) being positive and (b) summing up to one. For faster convergence, this update procedure can be interleaved with a line search method (e.g., Armijo's rule) to pick the step sizes at each iteration. These two steps are repeated until convergence, which can be checked by monitoring the successive objective function values.

Algorithm 1. Multitask Multiple Kernel Learning with Separate Parameters

```
1: Initialize $\boldsymbol{\eta}^r$ as $(1/P \ldots 1/P)^\top$  $\forall r$
2: repeat
3:     Calculate $\mathbf{K}_\eta^r = \{k_\eta^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r; \boldsymbol{\eta}^r)\}_{i,j=1}^{N^r}$  $\forall r$
4:     Solve a single-kernel machine using $\mathbf{K}_\eta^r$  $\forall r$
5:     Update $\boldsymbol{\eta}^r$ in the opposite direction of $\partial\mathcal{O}_\eta/\partial\boldsymbol{\eta}^r$  $\forall r$
6: until convergence
```

If the kernel combination parameters are regularized with the function (3), in the binary classification case, the gradients with respect to $\boldsymbol{\eta}^r$ are

$$\frac{\partial \mathcal{O}_\eta}{\partial \eta_m^r} = -2\nu \sum_{s=1}^{T} \eta_m^s - \frac{1}{2} \sum_{i=1}^{N^r} \sum_{j=1}^{N^r} \alpha_i^r \alpha_j^r y_i^r y_j^r k_m^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r)$$

and, in the regression case,

$$\frac{\partial \mathcal{O}_\eta}{\partial \eta_m^r} = -2\nu \sum_{s=1}^{T} \eta_m^s - \frac{1}{2} \sum_{i=1}^{N^r} \sum_{j=1}^{N^r} \alpha_i^r \alpha_j^r k_m^r(\boldsymbol{x}_i^r, \boldsymbol{x}_j^r).$$

4 Experiments

We test the proposed MTMKL algorithm on three data sets. We implement the algorithm and baseline methods, altogether one STMKL and three MTMKL algorithms, in MATLAB[1]. `STMKL` learns separate STMKL models for each task. `MTMKL(R)` is the MKL variant of regularized MTL model of [3], outlined in (1). `MTMKL(C)` is the MTMKL model that has common kernel combination parameters across the tasks, outlined in [9]. `MTMKL(S)` is the new MTMKL model that has separate but regularized kernel combination parameters across the tasks, outlined in Algorithm 1.

We use the squared error loss for both classification and regression problems. The regularization parameters C and ν are selected using cross-validation from $\{0.01, 0.1, 1, 10, 100\}$ and $\{0.0001, 0.01, 1, 100, 10000\}$, respectively. For each data set, we use the same cross-validation setting (i.e., the percentage of data used in training and the number of folds used for splitting the training data) reported in the previous studies to have directly comparable results.

4.1 Cross-Platform siRNA Efficacy Data Set

The cross-platform small interfering RNA (siRNA) efficacy data set[2] contains 653 siRNAs targeted on 52 genes from 14 cross-platform experiments with corresponding 19 features. We combine 19 linear kernels calculated on each feature separately. Each experiment is treated as a separate task and we use ten random splits where 80 per cent of the data is used for training. We apply two-fold cross-validation on the training data to choose regularization parameters.

Table 1. Root mean squared errors on the cross-platform siRNA data set

Method	RMSE
`STMKL`	23.89 ± 0.97
`MTMKL(R)`	37.66 ± 2.38
`MTMKL(C)`	23.53 ± 1.05
`MTMKL(S)`	23.45 ± 1.05

Table 1 gives the root mean squared error for each algorithm. `MTMKL(R)` is outperformed by all other algorithms because the target output spaces of the experiments are very different. Hence, training a separate learner for each cross-platform experiment is more reasonable. `MTMKL(C)` and `MTMKL(S)` are both better than `STMKL` in terms of the average performance, and `MTMKL(S)` is statistically significantly better (the paired t-test with the confidence level $\alpha = 0.05$).

[1] Implementations are available at `http://users.ics.tkk.fi/gonen/mtmkl`

[2] Available at `http://lifecenter.sgst.cn/RNAi`

4.2 MIT Letter Data Set

The MIT letter data set[3] contains 8×16 binary images of handwritten letters from over 180 different writers. A multitask learning problem, which has eight binary classification problems as its tasks, is constructed from the following pairs of letters and the number of data instances for each task is given in parentheses: `{a,g}` (6506), `{a,o}` (7931), `{c,e}` (7069), `{f,t}` (3057), `{g,y}` (3693), `{h,n}` (5886), `{i,j}` (5102), and `{m,n}` (6626). We combine five different kernels on binary feature vectors: the linear kernel and the polynomial kernel with degrees 2, 3, 4, and 5. We use ten random splits where 50 per cent of the data of each task is used for training. We apply three-fold cross-validation on the training data to choose regularization parameters. Note that `MTMKL(R)` cannot be trained for this problem because the output domains of the tasks are different.

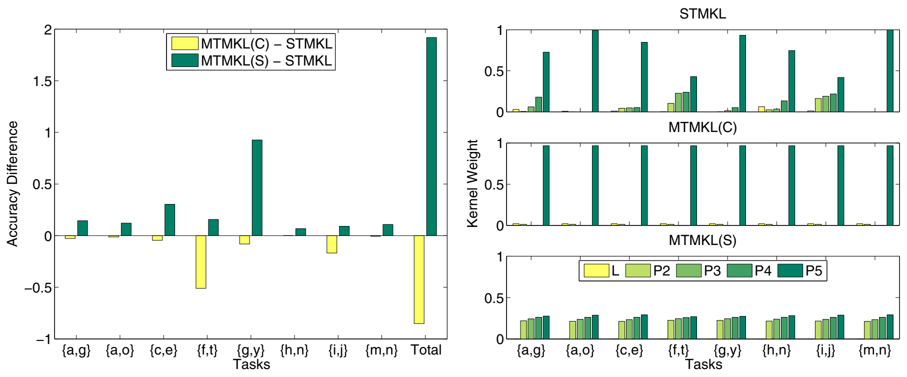

Fig. 1. Comparison of the three algorithms on the MIT letter data set. **Left:** Average accuracy differences. **Right:** Average kernel weights.

Figure 1 shows the average accuracy differences of `MTMKL(C)` and `MTMKL(S)` over `STMKL`. We see that `MTMKL(S)` consistently improves classification accuracy compared to `STMKL` and the improvement is statistically significant on six out of eights tasks (the paired t-test with the confidence level $\alpha = 0.05$), whereas `MTMKL(C)` could not improve classification accuracy on any of the tasks and it is statistically significantly worse on two tasks. Figure 1 also gives the average kernel weights of `STMKL`, `MTMKL(C)`, and `MTMKL(S)`. We see that `STMKL` and `MTMKL(C)` use the fifth degree polynomial kernel with very high weights, whereas `MTMKL(S)` uses all four polynomial kernels with nearly equal weights.

4.3 Cognitive State Inference Data Set

Finally, we evaluate the algorithms on a multilabel setting where each label is regarded as a task. The learning problem is to infer latent affective and cognitive states of a computer user based on physiological measurements. In the

[3] Available at `http://www.cis.upenn.edu/~taskar/ocr`

experiments, we measure six male users with four sensors (an accelerometer, a single-line EEG, an eye tracker, and a heart-rate sensor) while they are shown 35 web pages that include a personal survey, several preference questions, logic puzzles, feedback to their answers, and some instructions, one for each page. After the experiment, they are asked to annotate their cognitive state over three numerical Likert scales (valence, arousal, and cognitive load). Our features consist of summary measures of the sensor signals extracted from each page. Hence, our data set consisted of $6 \times 35 = 210$ data points and three output labels for each. We combine four Gaussian kernels on feature vectors of each sensor separately. We use ten random splits where 75 per cent of the data of each task is used for training. We apply three-fold cross-validation on the training data to choose regularization parameters. Note that `MTMKL(R)` cannot be applied to multilabel classification.

Learning inference models of this kind, which predict the cognitive and emotional state of the user, has a central role in cognitive user interface design. In such setups, a major challenge is that the training labels are inaccurate and scarce because collecting them is laborious to the users.

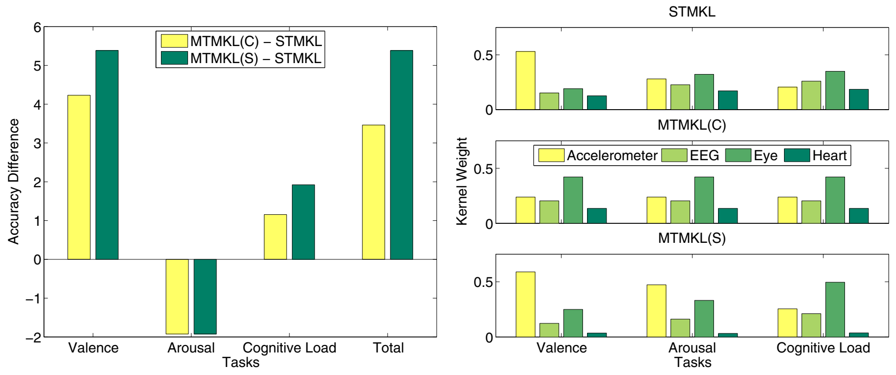

Fig. 2. Comparison of the three algorithms on the cognitive state inference data set. **Left:** Average accuracy differences. **Right:** Average kernel weights.

Figure 2 shows the accuracy differences of `MTMKL(C)` and `MTMKL(S)` over `STMKL` and reveals that learning and predicting the labels jointly helps to eliminate the noise present in the labels. Two of the three output labels (valence and cognitive load) are predicted more accurately in a multitask setup, with a positive change in the total accuracy. Note that `MTMKL(S)` is better than `MTMKL(C)` at predicting these two labels, and they perform equally well for the remaining one (arousal). Figure 2 also gives the kernel weights of `STMKL`, `MTMKL(C)`, and `MTMKL(S)`. We see that `STMKL` assigns very different weights to sensors for each label, whereas `MTMKL(C)` obtains better classification performance using the same weights across labels. `MTMKL(S)` assigns kernel weights between these two extremes and further increases the classification performance. We also see that the features extracted from the accelerometer are more informative than the other features for predicting valence; likewise, eye tracker is more informative for predicting cognitive load.

4.4 Computational Complexity

Table 2 summarizes the average running times of the algorithms on the data sets used. Note that `MTMKL(R)` and `MTMKL(S)` need to choose two parameters, C and ν, whereas `STMKL` and `MTMKL(C)` choose only C in the cross-validation phase. `MTMKL(R)` uses the training instances of all tasks in a single learner and always requires significantly more time than the other algorithms. We also see that `STMKL` and `MTMKL(C)` take comparable times and `MTMKL(S)` takes more time than these two because of the longer cross-validation phase.

Table 2. Running times of the algorithms in seconds

Data Set	`STMKL`	`MTMKL(R)`	`MTMKL(C)`	`MTMKL(S)`
Cross-Platform siRNA Efficacy	7.14	114.88	4.78	16.17
MIT Letter	9211.60	NA	8847.14	18241.32
Cognitive State Inference	5.23	NA	3.32	20.53

5 Conclusions

In this paper, we introduce a novel multiple kernel learning algorithm for multitask learning. The proposed algorithm uses separate kernel weights for each task, regularized to be similar. We show that training using a projected gradient-descent method is efficient. Defining the interaction between tasks to be over kernel weights instead of over other model parameters allows learning multitask models even when the input and/or output characteristics of the tasks are different. Empirical results on several data sets show that the proposed method provides high generalization performance with reasonable computational cost.

Acknowledgments. The authors belong to the Adaptive Informatics Research Centre (AIRC), a Center of Excellence of the Academy of Finland. This work was supported by the Nokia Research Center (NRC) and in part by the Pattern Analysis, Statistical Modeling and Computational Learning (PASCAL2), a Network of Excellence of the European Union.

References

1. Baxter, J.: A Bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning 28(1), 7–39 (1997)
2. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)

3. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 109–117. ACM (2004)
4. Gönen, M., Alpaydın, E.: Multiple kernel learning algorithms. Journal of Machine Learning Research 12, 2211–2268 (2011)
5. Ji, S., Sun, L., Jin, R., Ye, J.: Multi-label multiple kernel learning. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 21, pp. 777–784. MIT Press (2009)
6. Obozinski, G., Taskar, B., Jordan, M.I.: Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing 20(2), 231–252 (2009)
7. Parameswaran, S., Weinberger, K.Q.: Large margin multi-task metric learning. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 1867–1875. MIT (2010)
8. Rakotomamonjy, A., Bach, F.R., Canu, S., Grandvalet, Y.: SimpleMKL. Journal of Machine Learning Research 9, 2491–2521 (2008)
9. Rakotomamonjy, A., Flamary, R., Gasso, G., Canu, S.: $\ell_p - \ell_q$ penalty for sparse linear and sparse multiple kernel multi-task learning. IEEE Transactions on Neural Networks 22(8), 1307–1320 (2011)
10. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
11. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York (2004)
12. Tang, L., Chen, J., Ye, J.: On multiple kernel learning with multiple labels. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artifical Intelligence, pp. 1255–1260 (2009)
13. Varma, M., Babu, B.R.: More generality in efficient multiple kernel learning. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) Proceedings of the 26th International Conference on Machine Learning, p. 134. ACM (2009)
14. Xu, Z., Jin, R., Yang, H., King, I., Lyu, M.R.: Simple and efficient multiple kernel learning by group Lasso. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning, pp. 1175–1182. Omnipress (2010)