7-2013

MKBoost: A framework of multiple kernel boosting

Hao XIA
Nanyang Technological University

Steven C. H. HOI
Singapore Management University, chhoi@smu.edu.sg

MKBoost: A Framework of Multiple Kernel Boosting

Hao Xia* Steven C. H. Hoi†

Abstract

Multiple kernel learning (MKL) has been shown as a promising machine learning technique for data mining tasks by integrating with multiple diverse kernel functions. Traditional MKL methods often formulate the problem as an optimization task of learning both optimal combination of kernels and classifiers, and attempt to resolve the challenging optimization task by various techniques. Unlike the existing MKL methods, in this paper, we investigate a boosting framework of exploring multiple kernel learning for classification tasks. In particular, we present a novel framework of Multiple Kernel Boosting (MKBoost), which applies boosting techniques for learning kernel-based classifiers with multiple kernels. Based on the proposed framework, we develop several variants of MKBoost algorithms and examine their empirical performance in comparisons to several state-of-the-art MKL algorithms on classification tasks. Experimental results show that the proposed method is more effective and efficient than the existing MKL techniques.

1 Introduction

Kernel methods are a family of important techniques in data mining and machine learning for their proven state-of-the-art performance in many real data analysis applications [19, 20]. Most existing kernel methods usually adopt some kernel function $k(x_i, x_j)$ that computes the similarity between two data examples x_i and x_j. For many real-world scenarios, employing a single kernel function may not be enough since real data may come from multiple different sources or could be represented in different kinds of representations. This motivates the need of studying machine learning techniques for combining multiple kernels to achieve better capability and higher flexibility in solving real-world challenges. Such a learning problem is often known as "Multiple Kernel Learning" (MKL) in machine learning and data mining [21].

Recent years have witnessed a surge of studies that actively investigate various techniques for Multiple Kernel Learning [18, 21, 24]. Similar to some existing kernel methods, e.g. Support Vector Machines (SVM) [23], MKL has been demonstrated as a promising technique for solving many real-world applications, especially for its power of exploiting multiple kernels for fusing diverse information from multiple sources. For instance, MKL techniques have been applied to resolve the challenges of Protein subcellular localization in bioinformatics, speech recognition in signal processing, and anomaly detection in data mining [6].

Despite being studied extensively, the regular MKL methods have some critical limitations. In particular, most existing MKL methods often formulate the problem as a convex optimization task, e.g. a Semi-Definite Program (SDP) [16], and then attempt to resolve the optimization task by applying existing optimization techniques. Despite the nice convex formulation, it is often a great challenge for solving such complicated optimization tasks. Recently, a surge of research efforts have attempted to improve the efficiency of the optimization task by various techniques [21, 24]. For instance, the study in [21] formulated the MKL problem as a min-max optimization task, and proposed to find the saddle-point solution by solving a Semi-Infinite Linear Program (SILP). Although some encouraging progress has been made in this research direction, efficiency and scalability issues remain a key limitation for the existing regular MKL methods towards a real large application.

Besides the efficiency and scalability issues, the regular MKL methods usually assume a simple linear combination of multiple kernels, and attempt to learn a "flat" kernel-based classifier with the optimal linear combination of multiple kernels (similar to a regular SVM classifier). The classifier learned by such an approach is often "shallow", which may not fully exploit the power of multiple kernels for fitting diverse patterns.

To address the above limitations of the existing MKL methods, in this paper, we investigate a new multiple kernel learning framework, termed "Multiple Kernel Boosting" (MKBoost), which adapts the idea of boosting for learning classifiers with multiple kernels. The intuitive idea of multiple kernel boosting is to employ the boosting framework to learn an ensemble of multiple base kernel classifiers, each of them is equipped with one base kernel. The weights for both the kernel

*Nanyang Technological University, xiah0002@ntu.edu.sg
†Nanyang Technological University, chhoi@ntu.edu.sg

and classifier combination can be easily determined through the boosting process. As a result, we are able to efficiently learn a classifier with multiple kernels without resolving a complicated optimization task as required in a regular MKL approach.

As a summary, the contributions of this work include: (1) we present a novel framework of Multiple Kernel Boosting (MKBoost), which gives a new approach to learning kernel based classifiers with multiple kernels efficiently; (2) we propose two kinds of MKBoost algorithms: deterministic and stochastic algorithms, which attempt to resolve the MKBoost problem to trade off of both accuracy and efficiency; (3) we conduct extensive experiments for empirically validating the performance of the proposed MKBoost algorithms by comparing with the state-of-the-art MKL algorithms.

The rest of this paper will be organized as follows. Section 2 formulates the proposed framework of multiple kernel boosting, and gives two kinds of different approaches for solving the MKBoost problem. Section 3 discusses our empirical study. Section 4 reviews some related work, and Section 5 concludes this paper.

2 Multiple Kernel Boosting

In this section, we present a novel framework of multiple kernel boosting (MKBoost), which adapts boosting techniques for multiple kernel learning. Before presenting our MKBoost algorithms, we will first introduce the problem and review the regular multiple kernel learning (MKL). To ease our presentation, we will restrict our discussion on a typical binary classification task. Similar algorithms in this work can be easily extended to multi-class classification tasks.

2.1 Problem Formulation Consider a given set of training examples $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, N\}$ where $y_i \in \{-1, +1\}, i = 1, \ldots, N$, and a collection of M kernel functions $\mathcal{K} = \{\kappa_j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, j = 1, \ldots, M\}$.

The goal of our multiple kernel boosting task is to learn a kernel based classifier f, which is an ensemble of kernel classifiers using the collection of M kernel classifiers from the given training data examples. Typically, we can express such a kernel classifier as follows:

$$f(x_i) = \sum_{t=1}^{T} \alpha_t f_t(x_i) \tag{2.1}$$

where f_t is a hypothesis learned from a boosting trial, α_t is its associated weight in the final classifier, and T is the total number of boosting trials. The main challenge of multiple kernel boosting is how to develop an effective boosting scheme to learn the optimal hypothesis f_t and its combination weight α_t at each boosting trial.

For instance, if we consider only a single kernel κ, f_t can be learned by applying any regular kernel classifier such as SVM from a subset of n training examples in a boosting trial, i.e.,

$$f_t(\mathbf{x}) = \sum_{i=1}^{n} \beta_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) \tag{2.2}$$

where β_i's are the coefficients of an SVM model. Such an approach reduces to a regular boosting approach using a single kernel SVM as the base classifier.

In this paper, we present two kinds of approaches to solving the MKBoost problem to trade off both efficacy and efficiency. Before presenting our algorithms, we first review the basic formulation of regular MKL to better understand the motivation and difference of our work.

2.2 Multiple Kernel Learning The goal of a regular MKL task is to identify the optimal combination of M kernels, denoted by $\theta = (\theta_1, \ldots, \theta_M)$, which minimizes the margin based classification error. It can be cast into the following optimization problem:

$$\min_{\theta \in \Delta} \min_{f \in \mathcal{H}_{K(\theta)}} \frac{1}{2} |f|^2_{\mathcal{H}_{K(\theta)}} + C \sum_{i=1}^{N} \ell(f(x_i), y_i) \tag{2.3}$$

where $\Delta = \{\theta \in \mathbb{R}^M_+ | \theta^\top e_M = 1\}$, $K(\theta)(\cdot, \cdot) = \sum_{j=1}^{M} \theta_j \kappa_j(\cdot, \cdot)$, $\ell(f(x_i), y_i) = \max(0, 1 - y_i f(x_i))$.

In the above formulation, we use notation e_M to represent a vector of M dimensions with all its elements being 1. The above formulation can also be turned into the following min-max optimization problem:

$$\min_{\theta \in \Delta} \max_{\alpha \in \Xi} \left\{ \alpha^\top e_N - \tfrac{1}{2} (\alpha \circ \mathbf{y})^\top \left(\textstyle\sum_{j=1}^{M} \theta_j K^j \right) (\alpha \circ \mathbf{y}) \right\}$$

where $K^j \in \mathbb{R}^{N \times N}$ with $K^j_{p,q} = \kappa_i(x_p, x_q)$, $\Xi = \{\alpha | \alpha \in [0, C]^N\}$, and $\circ$ defines the element-wise product between two vectors. We refer to the above formulation as a regular MKL problem. Despite some encouraging results achieved recently [18, 24], developing an efficient and scalable algorithm for solving the regular MKL problem remains an open challenge.

2.3 Deterministic MKBoost Algorithms The general idea of MKBoost is to apply boosting techniques to learn a classifier using multiple kernels. To this purpose, we follow the typical procedure of a boosting algorithm, i.e., Adaptive Boosting known as "Adaboost" [7], which is a popular and successful boosting technique.

In particular, we repeatedly learn some kernel classifiers with multiple kernels f_t through a series of boosting trials $t = 1, \ldots, T$, where T denotes the total number of boosting trials. For each boosting trial, a distribution of weights D_t is engaged to indicate the importance

of the training examples for the classification. At each boosting trial, we increase the weights of the wrongly classified examples and/or decrease the weights of those correctly classified examples in order to focus on those examples that are hard to be correctly classified.

During each boosting trial, we first sample a subset of n training examples according to distribution D_t and a predefined *boosting sampling ratio* that determines the proportion of training data to be sampled at each boosting trial. Once obtaining the subset of training data, the next key issue is how to learn the kernel based classifier f_t from these training data. The first approach is to learn one classifier h_t^j with each kernel κ_j from the set of M kernels using a regular kernel method, e.g., SVM used in our study. Based on the set of M base classifiers, we can further measure the misclassification performance of each classifier h_t^j with kernel κ_j over distribution D_t of the whole collection of training data:

$$\epsilon_t^j = \epsilon(h_t^j) = \sum_{i=1}^{N} D_t(i)(h_t^j(x_i) \neq y_i) \tag{2.4}$$

As a result, we can build the classifier f_t for the t-th boosting trial by choosing the best classifier with the smallest misclassification rate, i.e.,

$$h_t = \arg \min_{j \in \{1,\ldots,M\}} \epsilon(h_t^j) \tag{2.5}$$

The next step is to follow the similar procedure of the Adaboost algorithm by computing the misclassification rate ϵ_t for the combined classifier h_t over the distribution D_t on the whole collection of training data:

$$\epsilon_t = \sum_{i=1}^{N} D_t(i)(h_t(x_i) \neq y_i) \tag{2.6}$$

The last step of each boosting trial is to update the weight of each training example $D_{t+1}(i)$ as follows:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if\ h_t(x_i) = y_i \\ e^{\alpha_t} & if\ h_t(x_i) \neq y_i \end{cases}$$

where $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$ and Z_t is a normalization factor to make D_{t+1} a distribution. Finally, after finishing all T boosting trials, the algorithm outputs the final classifier as follows:

$$h(x) = sign\Big(\sum_{t=1}^{T} \alpha_t h_t(x)\Big) \tag{2.7}$$

We refer to the above MKBoost algorithm as "MKBoost-D1" for short. The details of the proposed algorithm are shown in Algorithm 1.

Algorithm 1 The MKBoost algorithm (MKBoost-D1)

INPUT:
- training data: $(x_1, y_1), \ldots, (x_N, y_N)$
- kernel functions: $k_j(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, j = 1, \ldots, M$
- initial distribution $D_1(i) = 1/N, i = 1, \ldots, N$

for $t = 1, 2, \ldots, T$ **do**
  sample n examples using distribution D_t
  for $j = 1, 2, \ldots, M$ **do**
    train weak classifier with kernel k_j: $h_t^j : \mathcal{X} \to \{-1, +1\}$
    compute the training error over D_t: $\epsilon_t^j = \sum_{i=1}^{N} D_t(i)(h_t^j(x_i) \neq y_i)$
  end for
  select the best classifier with the minimal error rate

$$h_t = \arg\min_{h_t^j} \epsilon_t^j = \arg\min_{h_t^j} \sum_{i=1}^{N} D_t(i)(h_t^j(x_i) \neq y_i),$$

  choose $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$, where $\epsilon_t = \min_{j \in \{1,\ldots,M\}} \epsilon_t^j$
  Update $D_{t+1}(i)$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if\ h_t(x_i) = y_i \\ e^{\alpha_t} & if\ h_t(x_i) \neq y_i \end{cases}$$

  Z_t is a normalization factor to make D_t a distribution.
end for
OUTPUT: $h(x) = sign\big(\sum_{t=1}^{T} \alpha_t h_t(x)\big)$

Algorithm 2 The MKBoost Algorithm (MKBoost-D2)

INPUT:
- training data: $(x_1, y_1), \ldots, (x_N, y_N)$
- kernel functions: $k_j(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, j = 1, \ldots, M$
- initial distribution $D_1(i) = 1/N, i = 1, \ldots, N$

for $t = 1, 2, \ldots, T$ **do**
  sample a set of n examples using distribution D_t
  for $j = 1, 2, \ldots, M$ **do**
    train weak classifier with kernel k_j: $h_t^j : \mathcal{X} \to \{-1, +1\}$
    compute the training error over D_t: $\epsilon_t^j = \sum_{i=1}^{N} D_t(i)(h_t^j(x_i) \neq y_i)$
    choose $\alpha_t^j = \frac{1}{2} ln(\frac{1-\epsilon_t^j}{\epsilon_t^j})$
  end for
  combine the M classifiers: $h_t(x) = sign\big(\sum_{j=1}^{M} \alpha_t^j h_t^j(x)\big)$
  compute the training error over D_t: $\epsilon_t = \sum_{i=1}^{N} D_t(i)(h_t(x_i) \neq y_i)$
  choose $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
  Update $D_{t+1}(i)$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if\ h_t(x_i) = y_i \\ e^{\alpha_t} & if\ h_t(x_i) \neq y_i \end{cases}$$

  Z_t is a normalization factor to make D_t a distribution.
end for
OUTPUT: $h(x) = sign\big(\sum_{t=1}^{T} \alpha_t h_t(x)\big)$

In the above algorithm, we simply choose the best classifier among the M kernel classifiers as the classifier for the t-th boosting trial and the other $M-1$ classifiers are simply dropped. In some situation, it is possible that other kernel classifiers may make complementary contribution for improving classification performance. Thus, we propose another way to build the classifier by combining all these M classifiers, each of which is assigned with a combination weight. Specifically, we suggest to build the classifier for the t-th boosting trial:

$$h_t(x) = \text{sign}\Big(\sum_{j=1}^{M} \alpha_t^j h_t^j(x)\Big) \tag{2.8}$$

where the weight α_t^j is computed based on the misclassification rate ϵ_t^j, i.e.,

$$\alpha_t^j = \frac{1}{2}\ln\Big(\frac{1-\epsilon_t^j}{\epsilon_t^j}\Big) \tag{2.9}$$

We refer to this MKBoost algorithm as "MKBoost-D2" for short. The other part of this algorithm is same to the MKBoost-D1 algorithm. The details of the proposed MKBoost-D2 algorithm are shown in Algorithm 2.

2.4 Stochastic MKBoost Algorithms One important limitation for the above two MKBoost algorithms is that they need to repeatedly train a set of M kernel classifiers at each boosting trial. This could be computationally intensive if the number of kernels M is large in some applications. To address this limitation, in this section, we propose a *stochastic* learning approach for MKBoost, which aims to avoid the need of training all the M kernel classifiers. To ease our presentation, we refer to the new algorithms as *stochastic* MKBoost algorithms, and the previous two algorithms as *deterministic* MKBoost algorithms.

The intuitive idea of a stochastic MKBoost approach is that we could try to avoid unnecessary costs of training classifiers with some kernels that have relatively poor classification performance for the classification task. To this purpose, we introduce a variable $S_t(j)$ as the kernel sampling probability, which indicates how likely a kernel k_j will be sampled at the t-th boosting trial. At the beginning of the MKBoost algorithm, all $S_1(j)$ values are set to 1, which means that all M kernels will be sampled in the first boosting trial.

For each boosting trial, we sample a subset of kernels according to the kernel sampling probability S_t. The proposed MKBoost algorithm will train kernel classifiers only for those selected kernels. At the end of each boosting trial, we update the kernel sampling probability according to its classification performance:

$$S_{t+1}(j) \leftarrow S_t(j)\beta^{\epsilon_t^j} \tag{2.10}$$

Algorithm 3 The MKBoost algorithm (MKBoost-S1)

INPUT:
- training data: $(x_1, y_1), \ldots, (x_N, y_N)$
- kernel functions: $k_j(\cdot,\cdot) : \mathcal{X}\times\mathcal{X} \to \mathbb{R}, j = 1, \ldots, M$
- init. data distribution $D_1(i) = 1/N, i = 1, \ldots, N$
- init. kernel sampling prob. $S_1(j) = 1, j = 1, \ldots, M$
- sampling decay rate $0 < \beta < 1$

for $t = 1, 2, \ldots, T$ **do**
  Sample n examples using distribution D_t
  Sample a set of kernels $\mathcal{K}_t$ by sampling probability S_t
  for $k_j \in \mathcal{K}_t$ **do**
    Train weak classifier with kernel k_j: $h_t^j : \mathcal{X} \to \{-1, +1\}$
    Compute the training error over D_t: $\epsilon_t^j = \sum_{i=1}^N D_t(i)(h_t^j(x_i) \neq y_i)$
    Update $S_{t+1}(j) \leftarrow S_t(j)\beta^{\epsilon_t^j}$
  end for
  Update $S_{t+1}(j) \leftarrow S_{t+1}(j)/Z^S$, $Z^S = \max(S_{t+1})$
  $h_t = \arg\min_{\kappa_j \in \mathcal{K}_t} \epsilon(h_t^j) = \sum_{i=1}^N D_t(i)(h_t^j(x_i) \neq y_i)$
  Compute the training error over D_t: $\epsilon_t = \sum_{i=1}^N D_t(i)(h_t(x_i) \neq y_i)$
  $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
  Update $D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t}\exp(-\alpha_t y_i h_t(x_i))$
end for
OUTPUT: $h(x) = sign\big(\sum_{t=1}^T \alpha_t h_t(x)\big)$

Algorithm 4 The MKBoost algorithm (MKBoost-S2)

INPUT:
- training data: $(x_1, y_1), \ldots, (x_N, y_N)$
- kernel functions: $k_j(\cdot,\cdot) : \mathcal{X}\times\mathcal{X} \to \mathbb{R}, j = 1, \ldots, M$
- init. data distribution $D_1(i) = 1/N, i = 1, \ldots, N$
- init. kernel sampling prob. $S_1(j) = 1, j = 1, \ldots, M$
- sampling decay rate $0 < \beta < 1$

for $t = 1, 2, \ldots$ **do**
  Sample n examples using distribution D_t
  Sample a set of kernels $\mathcal{K}_t$ by sampling probability S_t
  for $k_j \in \mathcal{K}_t$ **do**
    Train weak classifier with kernel k_j: $h_t^j : \mathcal{X} \to \{-1, +1\}$
    Compute the training error over D_t: $\epsilon_t^j = \sum_{i=1}^N D_t(i)(h_t^j(x_i) \neq y_i)$
    Choose $\alpha_t^j = \frac{1}{2}\ln(\frac{1-\epsilon_t^j}{\epsilon_t^j})$
    Update $S_{t+1}(j) \leftarrow S_t(j)\beta^{\epsilon_t^j}$
  end for
  Update $S_{t+1}(j) \leftarrow S_{t+1}(j)/Z^S$, $Z^S = \max(S_{t+1})$
  $h_t(x) = sign\big(\sum_{k_j \in \mathcal{K}_t} \alpha_t^j h_t^j(x)\big)$
  Compute the training error over D_t: $\epsilon_t = \sum_{i=1}^N D_t(i)(h_t(x_i) \neq y_i)$
  Choose $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
  Update $D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t}\exp(-\alpha_t y_i h_t(x_i))$
end for
OUTPUT: $h(x) = sign\big(\sum_{t=1}^T \alpha_t h_t(x)\big)$

where $\beta \in (0,1)$ is a constant parameter introduced as a sampling decay factor for updating the kernel sampling probability, and ϵ_t^j is the misclassification rate of the kernel classifier with a selected kernel k_j. The above formula indicates the larger the misclassification rate, the more decay penalty will be applied to the kernel to reduce the chance of being sampled in the next trial.

By incorporating the above kernel sampling scheme, we suggest two stochastic MKBoost algorithms, MKBoost-S1 and MKBoost-S2, which are corresponding to the previous two deterministic MKBoost algorithms, respectively. The details of these two algorithms are shown in Algorithm 3 and 4.

3 Experiments

In this section, we conduct an extensive set of experiments to examine the classification performance of the proposed MKBoost algorithms by comparing with several state-of-the-art MKL techniques in literature.

3.1 Experimental Testbed We evaluate the performance of MKBoost algorithms for both binary and multiclass classification tasks. We randomly choose a number of publicly available datasets from web machine learning repositories, which can be downloaded from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. Table 1 shows the summary of the datasets in our experiments. Note that the sizes of these datasets are not very large because we want to compare our algorithms with other existing MKL algorithms, most of them are often not scalable and only applicable to relatively small datasets. In general, our algorithms are efficient and scalable for large datasets.

Table 1: Summary of the datasets used in the experiments (where C, N, D denote # classes, #examples, #features, respectively).

Abbr.	name	source	C	N	D
D1	balance-scale	UCI	2	576	4
D2	german.numer	Statlog	2	1000	24
D3	glass	UCI	2	214	10
D4	ionosphere	UCI	2	351	34
D5	monks1	UCI	2	556	6
D6	sonar	UCI	2	208	60
D7	svmguide3	CWH03a	2	1243	21
D8	wdbc	UCI	2	569	30
D9	segment	Statlog	7	2310	19
D10	vehicle	Statlog	4	846	18

3.2 Comparison Schemes In our experiments, we compare our MKBoost algorithms against several state-of-the-art MKL algorithms using different optimization techniques, including the following:

- MKL-SD: An MKL algorithm solved by a Subgradient Descent (SD) optimization method [18].
- MKL-SILP: An MKL algorithm solved by a Semi-Infinite Linear Programming (SILP) approach [21].
- MKL-Level: An MKL algorithm where the optimization solved by an extended level method [24].
- Lp-MKL: An extended MKL algorithm that generalizes the regular ℓ_1-norm MKL method to arbitrary ℓ_p-norm MKL [14].

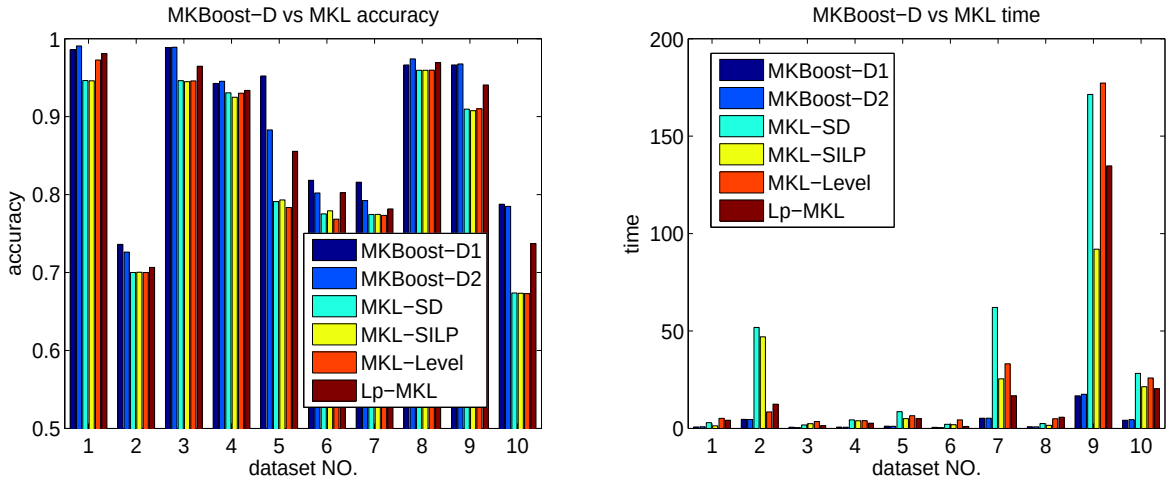
3.3 Experimental Setup For the setup of our experiments, we follow the typical approach used in the previous MKL studies in literature. In particular, for each dataset, we create a set of 17 base kernels, i.e.,

- Gaussian kernels with 14 different widths $(2^{-6}, 2^{-5}, \ldots, 2^7)$ on all features.
- Polynomial kernels of degree 1 to 3 on all features.

To implement the existing MKL algorithms, we adopt the SimpleMKL toolbox [18] that includes the implementations of both SILP and SD algorithms, and the LevelMKL toolbox [24] that implements MKL-Level. For running the above existing MKL algorithms, we adopt their default settings suggested by the two toolboxes, which employ the duality gap as the stopping criterion, and stop the optimization process when the duality gap is less than a threshold or the max number of optimization iterations is reached. For Lp-MKL, as no code available, we implemented it by ourselves, and set $p = 2$ for the best tradeoff of accuracy and efficiency.

For the implementation of our MKBoost algorithms, by default, we set the total number of boosting trials T to 100, the boosting sampling ratio to 0.2, and the sampling decay factor β to 2^{-5} for stochastic MKBoost algorithms. For SVM, we adopt the popular LIBSVM toolbox as the SVM solver. For the multiclass classification tasks, we adopt a one-to-one approach to training a set of binary classifiers. Finally, all the experiments were running in Matlab on a Linux machine with 3GHz Intel CPU and 16GB RAM.

For all the experiments, we repeat each algorithm 20 times on every dataset. Similar to the previous studies, for each run, 50% of the examples were randomly sampled as training data and the remaining were used for test. The training data were normalized to have zero mean and unit variance, and the test data were then normalized using the mean and variance of the training data. The penalty cost parameter C was fixed to 50 for SVM and the compared MKL algorithms. To avoid unstable results for stochastic MKBoost algorithms, we run 10 times under each setting and report average performances. Finally, we report both classification accuracy and time cost for performance evaluation.



(a) Accuracy (b) Time Cost (seconds)

Figure 1: Comparison between deterministic MKBoost algorithms and MKL algorithms.

Table 2: Comparison between two deterministic MKBoost algorithms and four different MKL algorithms.

Dataset	Metric	MKBoost-D1	MKBoost-D2	MKL-SD	MKL-SILP	MKL-Level	Lp-MKL
D1	accuracy	0.9858	**0.9907**	0.9464	0.9460	0.9726	0.9809
	std	±0.0079	**±0.0057**	±0.0099	±0.0132	±0.0102	±0.0078
	time	**0.7658**	0.8683	2.9520	1.3615	5.175	4.1735
D2	accuracy	**0.7361**	0.7263	0.7001	0.7003	0.7	0.7065
	std	**±0.0117**	±0.0094	±0.0004	±0.0025	±0.0000	±0.0055
	time	4.6168	**4.5845**	51.8245	46.9960	8.4915	12.43
D3	accuracy	0.9885	**0.9890**	0.9462	0.9448	0.9458	0.9646
	std	±0.0140	**±0.0121**	±0.0181	±0.0166	±0.0181	±0.0170
	time	0.6058	**0.4497**	1.812	2.4105	3.57	1.488
D4	accuracy	0.9426	**0.9453**	0.9306	0.9249	0.93	0.9337
	std	±0.0139	**±0.0103**	±0.0155	±0.0255	±0.0183	±0.0200
	time	0.697	**0.6048**	4.376	3.922	4.0235	2.683
D5	accuracy	**0.9522**	0.8829	0.791	0.7932	0.7835	0.8556
	std	**±0.0154**	±0.0199	±0.0251	±0.0311	±0.0232	±0.0221
	time	1.163	**1.0960**	8.6205	5.051	6.527	5.0955
D6	accuracy	**0.8183**	0.8021	0.7752	0.7791	0.7684	0.8024
	std	**±0.0396**	±0.0338	±0.0423	±0.0427	±0.0407	±0.0406
	time	0.5722	**0.4547**	2.135	1.9425	4.4125	1.012
D7	accuracy	**0.8158**	0.7923	0.7744	0.7746	0.7733	0.7816
	std	**±0.0067**	±0.0079	±0.0038	±0.0040	±0.0036	±0.0057
	time	5.2625	**5.2193**	62.118	25.4405	33.147	16.7775
D8	accuracy	0.9663	**0.9741**	0.9595	0.9595	0.9597	0.9692
	std	±0.0107	**±0.0074**	±0.0094	±0.0094	±0.0097	±0.0079
	time	0.8688	**0.8285**	2.514	1.576	4.9735	5.7715
D9	accuracy	0.9663	**0.9675**	0.9096	0.9076	0.9101	0.9406
	std	±0.0030	**±0.0026**	±0.0075	±0.0115	±0.0087	±0.0064
	time	**16.7547**	17.5432	171.386	91.966	177.194	134.704
D10	accuracy	**0.7877**	0.7848	0.6737	0.6733	0.673	0.7371
	std	**±0.0085**	±0.0115	±0.0191	±0.0199	±0.0186	±0.0140
	time	**4.1932**	4.5257	28.208	21.383	25.9065	20.429

3.4 Comparison Results We first compare the performances between two deterministic MKBoost algorithms (MKBoost-D1 and MKBoost-D2) and the four existing MKL algorithms (MKL-SD, MKL-SILP, MKL-Level, and Lp-MKL). Further, we will compare both the accuracy and efficiency performances between the deterministic and stochastic MKBoost algorithms. Finally, we will evaluate the effects of various parameters that may influence the performance of the MKBoost algorithms. To examine statistical significance of the comparisons, for the experimental results reported below, we highlight the best result in each group in bold font by conducting student t-tests with the significance level $\alpha = 0.05$.

3.4.1 MKBoost vs. Regular MKL Figure 1 and Table 2 show the experimental results of the comparisons between the two deterministic MKBoost algorithms and the four existing MKL algorithms.

Several observations can be drawn from the results. First of all, in terms of classification accuracy performance, among the three ℓ_1-norm MKL algorithms (MKL-SD, MKL-SILP, and MKL-Level), we found that their accuracy performances are generally comparable, in which no one algorithm significantly performs better than the others. This is consistent to the previous MKL studies as all these MKL algorithms are essentially based on the same formulation, but adopt different optimization techniques to speed up the optimization process. Further, comparing the three regular MKL algorithms and the two proposed MKBoost algorithms, we found the MKBoost algorithms considerably outperform the three regular MKL algorithms in most cases. For example, for dataset D3("glass"), the three regular MKL algorithms achieved the accuracy of no more than 95%, while the proposed deterministic MKBoost algorithms are able to attain a much higher accuracy of over 98% for both MKBoost-D1 and MKBoost-D2 algorithms. Moreover, by examining the performance of the state-of-the-art Lp-MKL algorithm, we found that it often performs better than the three regular ℓ_1-norm MKL algorithms, but still performs considerably worse than our MKBoost algorithms. Finally, by comparing the two deterministic MKBoost algorithms themselves, we found that their classification accuracy performances are generally comparable, in which MKBoost-D1 outperformed MKBoost-D2 in some datasets ("D2","D5","D6","D7","D10") but failed to beat MKBoost-D2 in the other datasets.

In addition to the clear gain of classification accuracy, we also found that the proposed MKBoost algorithms enjoy the significant advantage of higher learning efficiency over the regular MKL algorithms. In particular, by examining the time costs of learning the classification models, we found that the two proposed MKBoost algorithms are often significantly more efficient than the four existing MKL algorithms. Typically, both MKBoost algorithms are several times (about 2$\sim$10 times depending on datasets) faster than the existing MKL algorithms. For example, for the D2("german.number") dataset, the time costs of both MKBoost algorithms are only about one tenth of the costs by MKL-SD and MKL-SILP, and about half and one-third of the costs by MKL-Level and Lp-MKL, respectively. Finally, for the two proposed MKBoost algorithms, their time costs are comparable as they essentially share the same time complexity in theory.

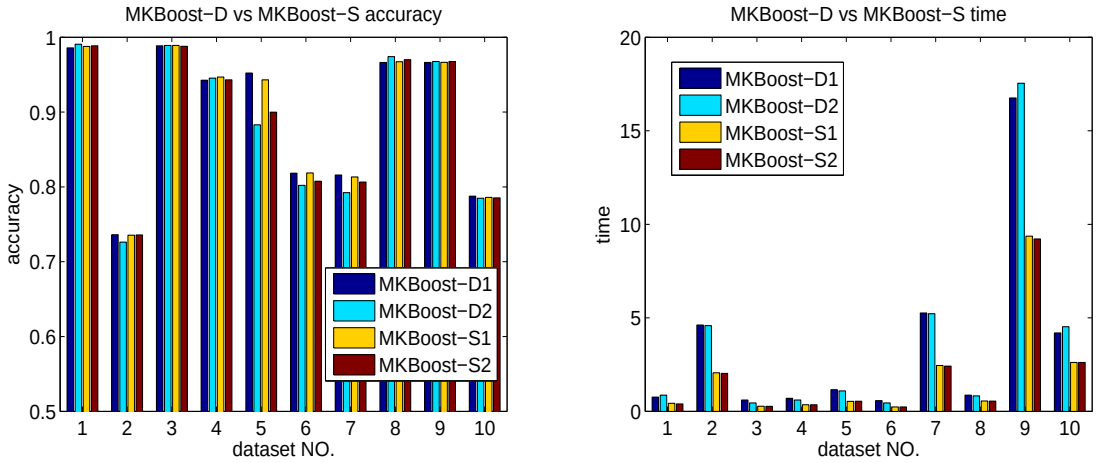
All the above encouraging experimental results show that the two MKBoost algorithms (MKBoost-D1 and MKBoost-D2) are not only more accurate than the regular MKL algorithms for the classification tasks, but also are able to learn the models considerably more efficiently.

3.4.2 Deterministic vs. Stochastic MKBoost Figure 2 and Table 3 show the experimental results of the comparisons between the two deterministic MKBoost algorithms (MKBoost-D1 and MKBoost-D2) and the two stochastic MKBoost algorithms (MKBoost-S1 and MKBoost-S2). Several observations can be drawn from the results.

First of all, it is clear to see that the stochastic MKBoost algorithms are more efficient than the deterministic MKBoost algorithms. The exact value of time efficiency speedup achieved by the stochastic algorithms depends on different datasets. Overall, the time cost of the two stochastic MKBoost algorithms is about half of the cost taken by the two deterministic algorithms as observed from this set of experiments.

In addition to the time efficiency issue, another concern is how accurate can the stochastic MKBoost algorithms perform in comparison to the deterministic algorithms. By further examining the classification accuracy results in details, it is a bit surprising to find that the two stochastic algorithms perform very well, which are in general fairly comparable to the deterministic algorithms. The difference of the accuracy values between a deterministic algorithm and a stochastic algorithm is often no more than 1%. Besides, for a few cases, we even observed that a stochastic algorithm slightly outperformed a deterministic algorithm. Finally, by comparing the two different stochastic algorithms, we found that both of their accuracy and efficiency performances are quite comparable, which is similar to the previous comparison of the two deterministic algorithms.

The above observations show that the proposed

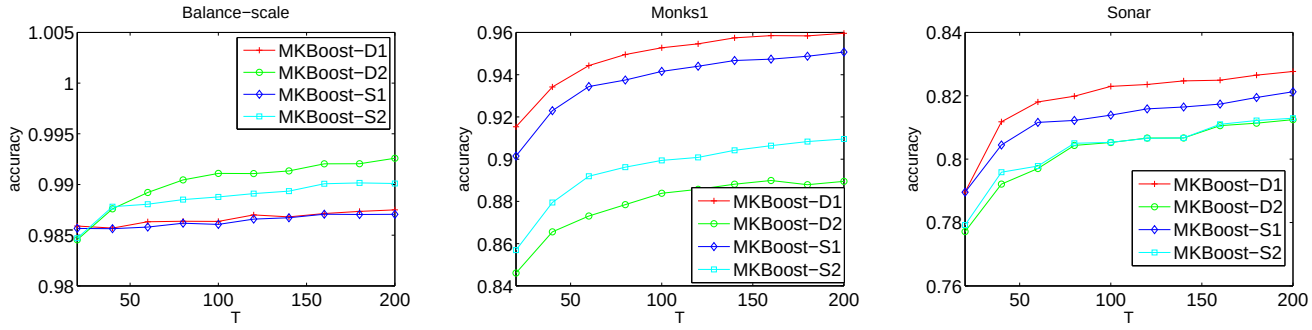


(a) Accuracy (b) Time Cost (seconds)

Figure 2: Comparison between deterministic and stochastic MKBoost algorithms.

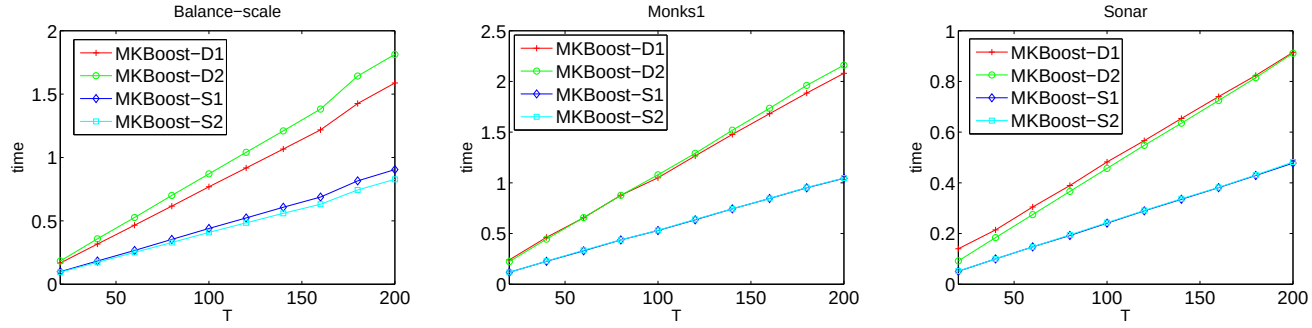
Table 3: Comparison between the deterministic and stochastic MKBoost algorithms.

Dataset	Metric	MKBoost-D1	MKBoost-D2	MKBoost-S1	MKBoost-S2
D1	accuracy	0.9858	**0.9907**	0.9878	0.9886
	std	±0.0079	±**0.0057**	±0.0067	±0.0061
	time	0.7658	0.8683	0.4350	**0.3975**
D2	accuracy	**0.7361**	0.7263	0.7355	0.7359
	std	±**0.0117**	±0.0094	±0.0106	±0.0079
	time	4.6168	4.5845	2.0623	**2.0320**
D3	accuracy	0.9885	**0.9890**	0.989	0.9879
	std	±0.0140	±**0.0121**	±0.0138	±0.0137
	time	0.6058	0.4497	0.2732	**0.2727**
D4	accuracy	0.9426	0.9453	**0.9469**	0.943
	std	±0.0139	±0.0103	±**0.0130**	±0.0152
	time	0.697	0.6048	0.3547	**0.3527**
D5	accuracy	**0.9522**	0.8829	0.943	0.8998
	std	±**0.0154**	±0.0199	±0.0153	±0.0221
	time	1.163	1.096	**0.5347**	0.5423
D6	accuracy	0.8183	0.8021	**0.8186**	0.8076
	std	±0.0396	±0.0338	±**0.0356**	±0.0338
	time	0.5722	0.4547	0.2388	**0.2370**
D7	accuracy	**0.8158**	0.7923	0.8132	0.8065
	std	±**0.0067**	±0.0079	±0.0066	±0.0080
	time	5.2625	5.2193	2.4548	**2.4192**
D8	accuracy	0.9663	**0.9741**	0.9673	0.9701
	std	±0.0107	±**0.0074**	±0.0102	±0.0087
	time	0.8688	0.8285	0.554	**0.5508**
D9	accuracy	0.9663	0.9675	0.9665	**0.9677**
	std	±0.0030	±0.0026	±0.0034	±**0.0028**
	time	16.7547	17.5432	9.366	**9.2138**
D10	accuracy	**0.7877**	0.7848	0.786	0.7853
	std	±**0.0085**	±0.0115	±0.0114	±0.0110
	time	4.1932	4.5257	2.6175	**2.6172**



(a) "Balance-scale" (b) "Monks1" (c) "Sonar"

Figure 3: Evaluation of MKBoost classification accuracy with respect to boosting parameter T.



(a) "Balance-scale" (b) "Monks1" (c) "Sonar"

Figure 4: Evaluation of MKBoost learning time cost with respect to boosting parameter T.

stochastic MKBoost algorithms can effectively speed up the deterministic algorithms by maintaining comparable classification accuracy.

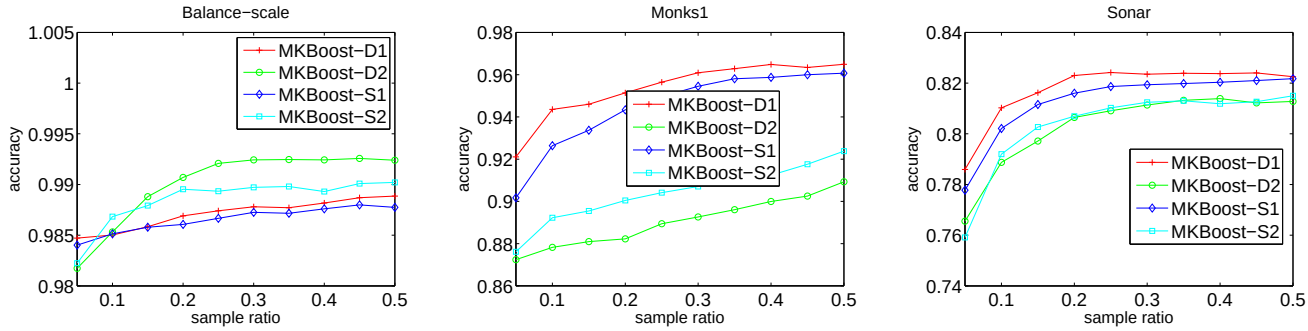
3.5 Parameter Evaluation We notice that there are a few parameters that could affect the performance (both accuracy and efficiency) of the proposed MKBoost algorithms. These parameters include the total number of boosting trials T, the sampling ratio of training data at each boosting trial, and the sampling decay factor β in the stochastic MKBoost algorithms. To examine the influence of these parameters, we conduct a set of experiments to evaluate their impacts on both accuracy and efficiency performance in the classification tasks.

3.5.1 Evaluation of Boosting Parameter T The first set of experiments is to examine the influence of the total number of boosting trials T for the proposed MKBoost algorithms. In our previous experiments, we simply fix this parameter to 100. In this set of experiments, we examine the experimental results by varying the parameter T from 20 to 200. Figure 3 and Figure 4 show the evaluation results for the impact of the boosting parameter T on the classification accuracy and learning time cost, respectively. Some observations can be drawn from the evaluation results.

First of all, we observed that, in terms of classification accuracy performance, increasing the total number of boosting trials T in general is able to boost the accuracy performance of all the proposed MKBoost algorithms consistently. Such observation is particularly more evident when the total number of boosting trials T is small (e.g. $T < 50$) at the beginning. The improvements of classification accuracy performance usually become very small when the parameter T is large (e.g. $T > 200$).

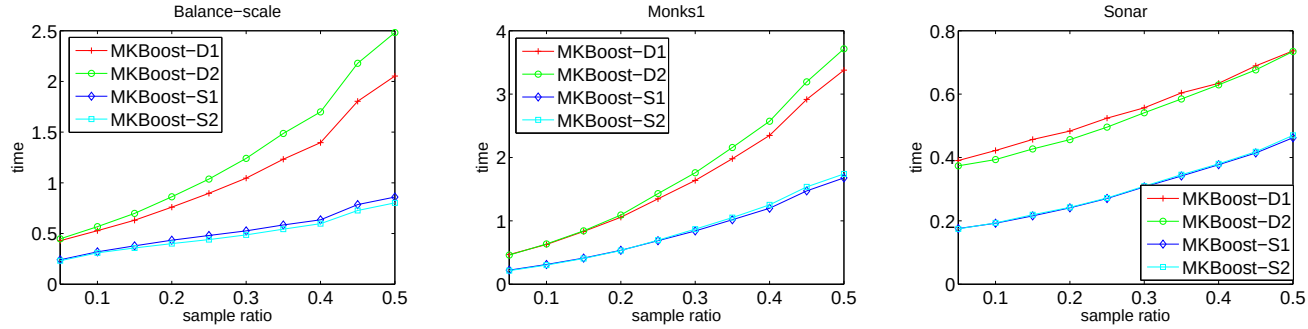
On the other hand, we found that increasing the value of T leads to the linear increase of the time cost required for training the models by all the proposed MKBoost algorithms. This is not surprising since all the proposed MKBoost algorithms have a linear time complexity with respect to the total number of boosting trials T.

The above empirical observations indicate that choosing an appropriate boosting parameter T is essentially a tradeoff between classification accuracy and efficiency performances. In practice, it is not difficult to choose an appropriate T value that usually falls in between 50 and 200, which sometimes also depends on the empirical requirements of efficiency and accuracy in a real-life application.



(a) "Balance-scale" (b) "Monks1" (c) "Sonar"

Figure 5: Evaluation of MKBoost classification accuracy with respect to boosting sampling ratio.



(a) "Balance-scale" (b) "Monks1" (c) "Sonar"

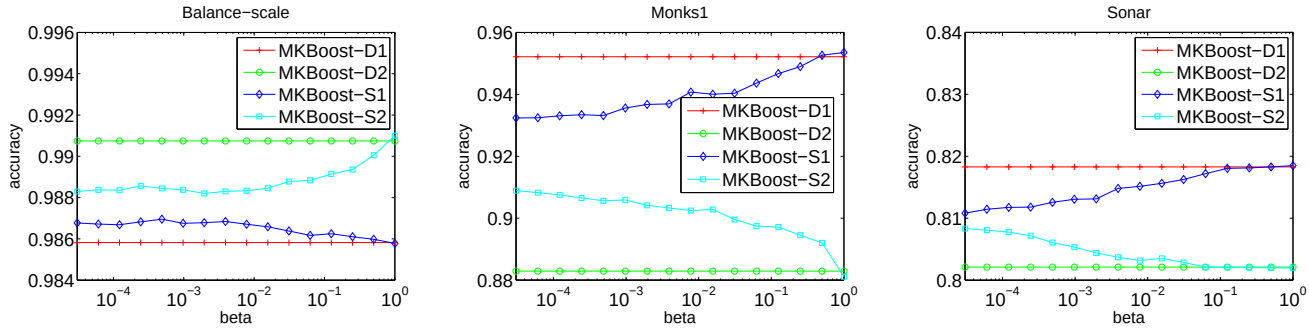
Figure 6: Evaluation of MKBoost learning time cost with respect to boosting sampling ratio.

3.5.2 Evaluation of Boosting Sampling Ratio Another boosting parameter that may affect the performance of the MKBoost algorithms is the boosting sampling ratio, which determines the proportion of training data examples sampled from the whole collection of training data at each boosting trial. Figure 5 and Figure 6 respectively show the evaluations of accuracy and efficiency performance with respect to the boosting sampling ratio by varying its value from 0.05 to 0.5.

From the experimental results, we found that the MKBoost algorithms with a large boosting sampling ratio value usually produced better classification accuracy performance. This is especially more evident when the boosting sampling ratio is small. For example, on the dataset "Monks1", the MKBoost-S1 algorithm has the accuracy of less than 93%, which was boosted to above 95% when the sampling ratio is increased to 0.3. Despite the improvement when increasing the boosting sampling ratio, we found that the classification accuracy tends to saturate when the value is large enough (e.g. larger than 0.4). All these observations are not difficult to interpret because when the sampling ratio is too small, the base kernel classifiers trained at the boosting process may suffer from insufficient training examples. On the other than, employing a too large sampling ratio may lead to sample too many training data examples for some large dataset, which may not be somewhat redundant for building the base classifiers at the boosting trials.

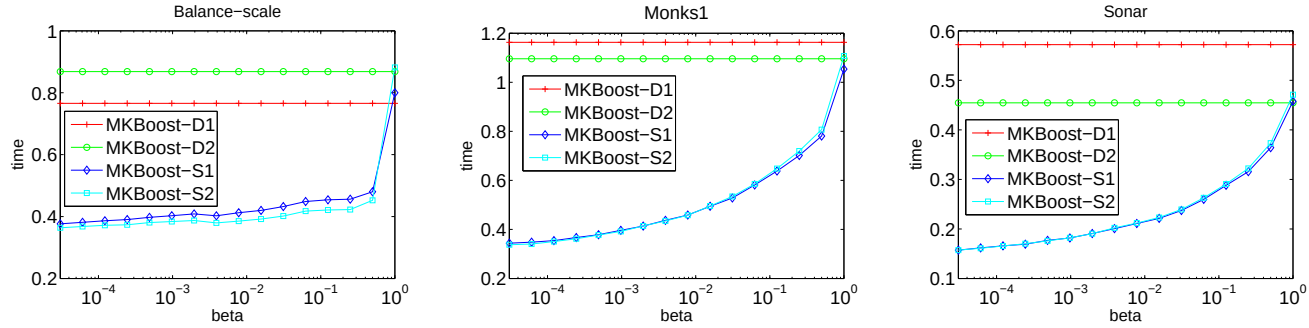
In addition to the impact on the accuracy performance, the boosting sampling ratio parameter also affects the time efficiency of the MKBoost algorithms. Similar to the observation from the evaluation of parameter T, increasing the boosting sampling ratio also leads to the increase of the time cost needed by the MKBoost algorithms. In particular, the relationship between the sampling ratio and the learning time cost of the MKBoost algorithms fully depends on the underlying algorithms used in training kernel based classifiers. As we adopt SVM for training the base kernel classifiers, the time cost of an MKBoost algorithm thus depends on the time cost of the SVM algorithm (e.g. the time complexity of the SMO algorithm implemented in the LIBSVM package is empirically in $O(n^{1.4})$ to $O(n^{2.3})$).

3.5.3 Evaluation of the Sampling Decay Factor β for Stochastic MKBoost Algorithms The last set of experiments is to examine the effect of the sampling decay factor β for the two stochastic MKBoost algorithms (MKBoost-S1 and MKBoost-S2). Figure 7 and Figure 8 show the evaluation of sampling decay factor in terms of classification accuracy and learning time cost performances, respectively. From the results,



(a) "Balance-scale" (b) "Monks1" (c) "Sonar"

Figure 7: Evaluation of accuracy of stochastic MKBoost algorithms with respect to the sampling decay factor β.



(a) "Balance-scale" (b) "Monks1" (c) "Sonar"

Figure 8: Evaluation of time cost of stochastic MKBoost algorithms with respect to the sampling decay factor β.

we found that when setting β to 1 in the extreme case, the two stochastic algorithms converge to the two respective deterministic algorithms, respectively. This is not surprising as $\beta = 1$ indicates that there is no decay at all for all the kernels, as a result the stochastic algorithm reduces the deterministic one. Further, we observed that when decreasing the value of β, the time cost can be consistently decreased since only some kernels will be sampled for training due to the decay effect. However, there is no a consistent trend for the impact on the accuracy performance by varying the β value. But in general we found that both stochastic MKBoost algorithms are fairly robust to the value of β, in which the accuracy results of the two stochastic algorithms using varied β values are usually in between the results of the two deterministic algorithms.

4 Related Work

Kernel methods have been extensively studied in literature [20, 5]. Most kernel methods usually assume some predefined parametric kernels, e.g. a gaussian kernel, is given a priori, where the parameters of these kernel functions are usually determined empirically by cross validation. Several studies have been proposed to learn parametric, semi-parametric or nonparametric kernel functions/matrices from labeled and/or unlabeled data. Example techniques include cluster kernels [3], diffusion kernels [15], marginalized kernels [13], graph-based spectral kernel learning methods [25, 10, 2], nonparametric kernel learning [9, 26] and so on.

Recently, Multiple Kernel Learning (MKL) [16, 16, 21, 17, 24, 12] has been actively studied, which aims to learn kernel based models by finding the optimal combination of a set of predefined kernels for a classification task. Example algorithms include MKL by the SDP approach [16], MKL by the SILP approach [21], MKL using the subgradient descent approach [18], and MKL by the level based optimization [24]. Recently, some emerging works also attempt to improve the regular MKL. For example, the ℓ_p-norm MKL method extended the regular ℓ_1-norm MKL for arbitrary ℓ_p-norm MKL with $p > 1$. Besides, some recent studies [27, 11, 8, 22] also attempted to address other issues of MKL, such as multi-class and multi-labeled classification. Unlike the existing MKL methods that often have to solve a complicated optimization task for combining multiple kernel classifiers, our work learns to combine multiple kernel classifiers using a boosting approach, which is more efficient and scalable than the regular MKL approaches. Finally, we note that our work is also related to some existing works that employ boosting for learning mixture-of-kernel models [1] or kernel design[4].

5 Conclusions

This paper presented a novel framework of multiple kernel boosting (MKBoost) for classification with multiple kernels. Unlike regular MKL methods that often have to solve challenging optimization tasks with multiple kernels, the proposed MKBoost technique is simple and easy to learn effective classification models with multiple kernels by exploiting the advantage of efficient boosting techniques. To further trade off between accuracy and efficiency in a classification task, we first proposed two deterministic MKBoost algorithms to train a set of kernel-based classifiers using all kernels at every boosting trial; we then presented two stochastic MKBoost algorithms that randomly sample a subset of kernels at a boosting trial. We found that MKBoost empirically achieved better accuracy than the regular MKL algorithms, and performed much more efficiently than the state-of-the-art MKL techniques. Besides, we also found that the stochastic MKBoost algorithms considerably improved the efficiency of the deterministic algorithms by maintaining comparable accuracy. For future work, an open problem is to theoretically analyze the generalization performance of the MKBoost algorithms.

acknowledgements

This work was supported in part by Singapore NRF Interactive Digital Media R&D Program under research grant (NRF2008IDM-IDM-004-018) and MOE ARC Tier-2 research grant (T208B2203).

References

[1] J. Bi, T. Zhang, and K.P. Bennett. Column-generation boosting methods for mixture of kernels. In *KDD*, page 526, 2004.

[2] Olivier Bousquet and Daniel J. L. Herrmann. On the complexity of learning the kernel matrix. In *NIPS*, pages 399–406, 2002.

[3] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In *NIPS*, pages 585–592, 2002.

[4] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *NIPS*, pages 537–544, 2002.

[5] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.

[6] Santanu Das, Bryan L. Matthews, Ashok N. Srivastava, and Nikunj C. Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *KDD'10*, pages 47–56, Washington, DC, USA, 2010.

[7] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[8] Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *ICML*, pages 352–359, 2008.

[9] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *ICML*, pages 361–368, 2007.

[10] Steven C. H. Hoi, Michael R. Lyu, and Edward Y. Chang. Learning the unified kernel machines for classification. In *KDD*, pages 187–196, 2006.

[11] Shuiwang Ji, Liang Sun, Rong Jin, and Jieping Ye. Multi-label multiple kernel learning. In *NIPS*, 2008.

[12] Rong Jin, Steven C. H. Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *ALT*, pages 390–404, 2010.

[13] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *ICML*, pages 321–328, 2003.

[14] Marius Kloft, Ulf Brefeld, Soeren Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate ℓ_p-norm multiple kernel learning. In *NIPS*, pages 997–1005, 2009.

[15] Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, pages 315–322, 2002.

[16] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

[17] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *ICML*, pages 775–782, 2007.

[18] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu, and Yves Grandvalet. Simplemkl. *JMLR*, 11:2491–2521, 2008.

[19] B. Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[20] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[21] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2006.

[22] Lei Tang, Jianhui Chen, and Jieping Ye. On multiple kernel learning with multiple labels. In *IJCAI*, pages 1255–1260, 2009.

[23] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[24] Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, 2008.

[25] Xiaojin Zhu, Jaz S. Kandola, Zoubin Ghahramani, and John D. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *NIPS*, 2004.

[26] Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. Simplenpkl: simple non-parametric kernel learning. In *ICML*, 2009.

[27] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *ICML*, pages 1191–1198, Corvalis, Oregon, 2007.