Noname manuscript No.
(will be inserted by the editor)

Online Multiple Kernel Classification

Steven C.H. Hoi, Rong Jin,
Peilin Zhao, Tianbao Yang

Received: Oct 25, 2010 / Accepted: Jul 19, 2012

Abstract Although both *online learning* and *kernel learning* have been studied extensively in machine learning, there is limited effort in addressing the intersecting research problems of these two important topics. As an attempt to fill the gap, we address a new research problem, termed **Online Multiple Kernel Classification** (OMKC), which learns a kernel-based prediction function by selecting a subset of predefined kernel functions in an online learning fashion. OMKC is in general more challenging than typical online learning because both the kernel classifiers and the subset of selected kernels are unknown, and more importantly the solutions to the kernel classifiers and their combination weights are correlated. The proposed algorithms are based on the fusion of two online learning algorithms, i.e., the *Perceptron* algorithm that learns a classifier for a given kernel, and the *Hedge* algorithm that combines classifiers by linear weights. We develop stochastic selection strategies that randomly select a subset of kernels for combination and model updating, thus improving the learning efficiency. Our empirical study with 15 data sets shows promising performance of the proposed algorithms for OMKC in both learning efficiency and prediction accuracy.

Keywords Online Learning · Kernel Methods · Multiple Kernels · Perceptron · Hedge · Classification

1 Introduction

In machine learning, online learning and kernel learning are two active research topics, which have been studied separately for years. Online learning is designed to sequentially learn a prediction model based on the feedback of answers to previous questions and possibly additional side information (Shalev-Shwartz 2007). It distinguishes from typical supervised learning algorithms that are designed to learn a classification model from a collection of given training examples. Kernel learning aims to learn an effective kernel function

S.C.H. Hoi · P. Zhao
School of Computer Engineering, Nanyang Technological University, Singapore 639798.
E-mail: chhoi@ntu.edu.sg, ZHAO0106@ntu.edu.sg

R. Jin · T. Yang
Department of Computer Science and Engineering, Michigan State University, USA.
E-mail: rongjin@cse.msu.edu, yangtia1@cse.msu.edu

for a given learning task from training data (Lanckriet et al. 2004; Sonnenburg et al. 2006; Hoi et al. 2007). An example of kernel learning is Multiple Kernel Learning (MKL) (Bach et al. 2004; Sonnenburg et al. 2006), which finds the optimal combination of multiple kernels to optimize the performance of kernel based learning methods.

Among various existing algorithms proposed for online learning (Freund and Schapire 1999; Crammer et al. 2006), several studies are devoted to examining kernel techniques in online learning settings (Freund and Schapire 1999; Crammer et al. 2006; Kivinen et al. 2001; Jyrki Kivinen and Williamson. 2004). However, most of the existing kernel based online learning algorithms assume that the kernel function is given a priori, significantly limiting their applications to real-world problems. As an attempt to overcome this limitation, we introduce a new research problem, **Online Multiple Kernel Classification** (OMKC), which aims to learn multiple kernel classifiers and their linear combination simultaneously. The main challenge arising from OMKC is that both the optimal kernel classifiers and their linear combinations need to be *online* learned simultaneously. More importantly, the solutions to kernel classifiers and their linear combinations are strongly correlated, making it a significantly more challenging problem than a typical online learning problem.

To this end, we propose a novel OMKC framework for online learning with multiple kernels, which fuses two kinds of online learning techniques: the *Perceptron* algorithm (Rosenblatt 1958) that learns a classifier for a given kernel, and the *Hedge* algorithm (Freund and Schapire 1997) that linearly combines multiple classifiers. We further develop kernel selection strategies that randomly choose a subset of kernels for model updating and combination, thus improving the learning efficiency significantly. We analyze the mistake bounds for the proposed OMKC algorithms. Our empirical studies with 15 datasets show promising performance of the proposed OMKC algorithms compared to the state-of-the-art algorithms for online kernel learning.

The rest of this paper is organized as follows. Section 2 reviews the related work in online learning and kernel learning. Section 3 defines the problem of online learning over multiple kernels and presents two different types of algorithms. Section 5 presents our empirical studies that extensively evaluates the performance of the proposed OMKC algorithms. Section 6 discusses some open issues and future directions. Section 7 concludes this paper.

2 Related Work

This section briefly reviews some major related work of online learning and kernel learning.

2.1 Online Learning

Recent years have witnessed a variety of online learning algorithms proposed and studied in different contexts and applications (Crammer et al. 2006; Yang, Xu, King and Lyu 2010). For more references, please kindly refer to the overview of online learning in Shalev-Shwartz (2007); Cesa-Bianchi and Lugosi (2006) and references therein.

A large number of recent studies in online learning are based on the framework of maximum margin learning. Most of these algorithms either extended or enhanced the well-known *Perceptron* algorithm (Agmon 1954; Rosenblatt 1958; Novikoff 1962), a pioneering online learning algorithm for linear prediction models. Exemplar algorithms in this category include the Relaxed Online Maximum Margin Algorithm (ROMMA) (Li and Long 2002), the Approximate Maximal Margin Classification Algorithm (ALMA) (Gentile 2001), the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer 2003), the NORMA

algorithm (Kivinen et al. 2001; Jyrki Kivinen and Williamson. 2004), the online Passive-Aggressive (PA) algorithms (Crammer et al. 2006), and the Double Updating Online Learning (DUOL) (Zhao et al. 2011), and the recent family of confidence-weighted learning algorithms (Dredze and Crammer 2008; Wang et al. 2012). Among them, several studies introduce kernel functions into online learning to achieve nonlinear classification (Kivinen et al. 2001; Freund and Schapire 1999; Crammer et al. 2006; Zhao et al. 2011). Similar to these studies, our OMKC framework is also a kernel based approach for online learning.

Moreover, some online learning studies mainly concern the budget issue, i.e., online learning with budget (Crammer et al. 2003; Cavallanti et al. 2007), which has received much interest recently. It differs from typical online learning methods in that the number of support vectors is bounded in training the classification models. Example algorithms include the first kind of approach to overcoming the unlimited growth of the support set (Crammer et al. 2003), the shifting Perceptron (Cavallanti et al. 2007), the Forgetron (Dekel et al. 2008), the Projectron (Orabona et al. 2008), and the very recent bounded online gradient descent algorithms (Zhao et al. 2012), etc.

In addition to the above online learning studies, our work is also related to online prediction with expert advice (Freund and Schapire 1997; Littlestone and Warmuth 1989, 1994; Vovk 1998). The most well-known and successful work is probably the Hedge algorithm (Freund and Schapire 1997), which was a direct generalization of Littlestone and Warmuth's Weighted Majority (WM) algorithm (Littlestone and Warmuth 1989, 1994). Other recent studies include the improved theoretical bounds (Cesa-Bianchi et al. 2007) and the parameter-free hedging algorithm (Chaudhuri et al. 2009) for decision-theoretic online learning. We refer readers to the book (Cesa-Bianchi and Lugosi 2006) for the other in-depth discussion of this subject.

2.2 Kernel Learning

How to find an effective kernel for a given task is critical to most kernel based methods in machine learning (Shawe-Taylor and Cristianini 2004; Cristianini et al. 2001). Most kernel methods assume that a predefined parametric kernel, e.g. a polynomial kernel or an RBF kernel, is given a priori and the parameters of these kernel functions are usually determined empirically by cross validation. Several studies proposed to learn parametric or semi-parametric kernel functions/matrices from labeled and/or unlabeled data. Exemplar techniques include cluster kernels (Chapelle et al. 2002), diffusion kernels (Kondor and Lafferty 2002), marginalized kernels (Kashima et al. 2003), idealized kernel learning (Kwok and Tsang 2003), and graph-based spectral kernel learning approaches (Zhu et al. 2004; Hoi et al. 2006; Bousquet and Herrmann 2002).

Another form of kernel learning, known as Multiple Kernel Learning (MKL) (Lanckriet et al. 2004), aims to find the optimal combination of multiple kernels for a classification task. Exemplar algorithms include the convex optimization (Lanckriet et al. 2004), the semi-infinite linear program (SILP) approach (Sonnenburg et al. 2006), the subgradient descent approach (Rakotomamonjy et al. 2008), and the level method (Xu et al. 2008). In addition, several recent studies (Zien and Ong 2007; Ji et al. 2008; Tang et al. 2009) addressed other MKL problems, such as MKL on multi-class and multi-labeled data, the compositional kernel combination method (Lee et al. 2007), multi-layer MKL (Zhuang, Tsang and Hoi 2011b), and unsupervised MKL (Zhuang, Wang, Hoi and Lan 2011). Our work differs from the existing MKL methods in that our goal is to resolve online classification tasks while most existing MKL methods were developed to mainly tackle batch classification tasks.

Besides learning kernels from labeled examples, several studies addressed the challenge of learning kernel matrices from side information (e.g., pairwise constraints). Methods in this category include nonparametric kernel learning (Hoi et al. 2007; Zhuang et al. 2009; Chen et al. 2009; Zhuang, Tsang and Hoi 2011a), lower-rank kernel learning (Kulis et al. 2006, 2009), generalized maximum entropy models (Yang, Jin and Jain 2010), and indefinite kernel learning (Chen and Ye 2008, 2009). Finally, there are some emerging studies for online multiple kernel learning (Jie et al. 2010; Martins et al. 2011) that address some other issues such as multi-class learning or structured prediction. We note that these studies might have been developed in parallel or after our earlier conference paper published in Jin et al. (2010). Our work differs from them in that we focus on enhancing online classification performance by choosing and combining multiple kernels, namely only a subset of kernels are selected for updating and combining during online learning process. It is the kernel selection strategies developed in this work that make the proposed learning algorithms significantly more efficient than the existing approaches for online multiple kernel learning.

3 Proposed Framework for Online Classification with Multiple Kernels

We introduce the problem setting and regular Multiple Kernel Learning (MKL), and then present the proposed framework of online multiple kernel classification.

3.1 Problem Setting and Multiple Kernel Learning

Consider a set of training examples $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, n\}$ where $x_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}, i = 1, \ldots, n$, and a collection of m kernel functions $\mathcal{K} = \{\kappa_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m\}$. The goal of multiple kernel learning is to learn a kernel-based prediction function by identifying the optimal combination of the m kernels, denoted by $\theta = (\theta_1, \ldots, \theta_m)$ to minimize the margin-based classification error. It is cast into the optimization below:

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

where

$$\Delta = \{\theta \in \mathbb{R}^m_+ | \theta^\top \mathbf{1}_m = 1\},\ K(\theta)(\cdot, \cdot) = \sum_{i=1}^{m} \theta_i \kappa_i(\cdot, \cdot),\ \ell(f(x_i), y_i) = \max(0, 1 - y_i f(x_i))$$

In the above formulation, we use notation $\mathbf{1}_m$ to represent a vector of m dimensions with all its elements being 1. It can also be cast into the following mini-max optimization problem:

$$\min_{\theta \in \Delta} \max_{\alpha \in \Xi} \left\{ \alpha^\top \mathbf{1}_n - \frac{1}{2}(\alpha \circ \mathbf{y})^\top \left(\sum_{i=1}^{m} \theta_i K^i \right) (\alpha \circ \mathbf{y}) \right\} \tag{2}$$

where $K^i \in \mathbb{R}^{n \times n}$ with $K^i_{j,l} = \kappa_i(x_j, x_l)$, $\Xi = \{\alpha | \alpha \in [0, C]^n\}$, and $\circ$ defines the element-wise product between two vectors. We refer to the above formulation as a regular batch MKL problem. Despite some encouraging results achieved recently (Rakotomamonjy et al. 2008; Xu et al. 2008), developing an efficient and scalable MKL algorithm remains an open research challenge in order to solve the challenging optimization task. Unlike the recent efforts for online MKL studies (Jie et al. 2010; Martins et al. 2011) which are mainly concerned in optimizing the optimal kernel combination, in this paper, we present a new framework for online multiple kernel classification which is focused on exploring effective online combination of multiple kernel classifiers.

3.2 The Proposed Framework of Online Multiple Kernel Classification

The proposed Online Multiple Kernel Classification (OMKC) framework is based on the fusion of two online learning methods: the Perceptron algorithm (Rosenblatt 1958) and the Hedge algorithm (Freund and Schapire 1997). In particular, for each kernel, the Perceptron algorithm is employed to learn a kernel-based classifier with some selected kernel, and the Hedge algorithm is used to update their combination weights. Algorithm 1 shows the detailed steps of the proposed framework.

Algorithm 1 Deterministic Algorithm for OMKC (**OMKC**$_{(\mathrm{D,D})}$)

1: INPUT:
   – Kernels: $k_i(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m$
   – Weights $w_i(1) = 1, i = 1, \ldots, m$
   – Discount weight $\beta \in (0, 1)$.
2: **Initialization**: $\mathbf{f}^1 = \mathbf{0}, \mathbf{w}^1 = \mathbf{1}, \boldsymbol{\theta}^1 = \frac{1}{m}\mathbf{1}$
3: **for** $t = 1, 2, \ldots$ **do**
4: Receive an instance: x_t
5: Predict $\hat{y}_t = \mathrm{sign}\Big(\sum_{i=1}^m \theta_i \mathrm{sign}\big(f_i^t(x_t)\big)\Big)$
6: Receive the class label: y_t
7: **for** $i = 1, 2, \ldots, m$ **do**
8: Set $z_i^t = I(y_t f_i^t(x_t) \leq 0)$
9: Update $w_i^{t+1} = w_i(t)\beta^{z_i^t}$
10: Update $f_i^{t+1}(x) = f_i^t(x) + z_i^t y_t \kappa_i(x_t, x)$
11: **end for**
12: $\theta_i^{t+1} = \frac{w_i^t}{W_t}, i = 1, \ldots, m$, where $W_t = \sum_{i=1}^m w_i^t$
13: **end for**

In this framework, we use $w_i(t)$ to denote the combination weight for the i-th kernel classifier at round t, which is set to 1 at the initial round. For each learning round, we update the weight $w_i(t)$ by following the Hedge algorithm as follows:

$$w_{t+1}^i = w_t^i \beta^{z_t^i}$$

where $\beta \in (0, 1)$ is a discount weight parameter, which is employed to penalize the kernel classifier that performs incorrect prediction at each learning step, and $z_i(t)$ indicates if the i-th kernel classifier makes a mistake on the prediction of the example x_t.

Next we derive a theorem to show the mistake bound for Algorithm 1. Throughout this paper, we assume $\kappa(x, x) \leq 1$ for any x. For the convenience of following discussions, we define the following notations:

$$W_t = \sum_{i=1}^m w_i^t,\ \theta_i^t = \frac{w_i^t}{W_t},\ z_i^t = I\left(y_t f_i^t(x_t) \leq 0\right),\ \hat{y}_t = \mathrm{sign}\Big(\sum_{i=1}^m \theta_i^t \mathrm{sign}\big(f_i^t(x_t)\big)\Big)$$

where $f_i^t(x)$ is used to represent the classifier at trial t that is constructed by using the kernel function $\kappa_i(\cdot,\cdot)$, and $I(x)$ is an indicator function that outputs 1 when x is true and 0 otherwise. Here, θ_i^t essentially defines the mixture of kernel classifiers, and z_i^t indicates if training example (x_t, y_t) is misclassified by the ith kernel classifier at trial t. Finally, we define the optimal margin classification error for the kernel $\kappa_i(\cdot,\cdot)$ with respect to a

collection of training examples $\mathcal{L} = \{(x_t, y_t), t = 1, \ldots, T\}$ as

$$F(\kappa_i, \ell, \mathcal{L}) = \min_{f \in \mathcal{H}_{\kappa_i}} \left(|f|^2_{\mathcal{H}_{\kappa_i}} + 2 \sum_{t=1}^{T} \ell(y_t, f(x_t)) \right) \tag{3}$$

Theorem 1 *After receiving a sequence of T training examples, denoted by $\mathcal{L} = \{(x_t, y_t), t = 1, \ldots, T\}$, the number of mistakes M made by running Algorithm 1, denoted by*

$$M = \sum_{t=1}^{T} I(y_t \hat{y}_t \leq 0) = \sum_{t=1}^{T} I\left(\sum_{i=1}^{m} \theta_i^t z_i^t \geq 0.5 \right)$$

is bounded as follows

$$M \leq \frac{2\ln(1/\beta)}{1-\beta} \min_{1 \leq i \leq m} \sum_{t=1}^{T} z_i^t + \frac{2 \ln m}{1-\beta} \tag{4}$$

$$\leq \frac{2\ln(1/\beta)}{1-\beta} \min_{1 \leq i \leq m} F(\kappa_i, \ell, \mathcal{L}) + \frac{2 \ln m}{1-\beta} \tag{5}$$

By choosing $\beta = \frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$, we have

$$M \leq 2 \left(\left(1 + \sqrt{\frac{\ln m}{T}} \right) \min_{1 \leq i \leq m} F(\kappa_i, \ell, \mathcal{L}) + \ln m + \sqrt{T \ln m} \right)$$

The proof of this theorem essentially combines the proof of the Perceptron algorithm and the Hedge algorithm. The details can be found in the Appendix. We note that the mistake bound in above theorem can be improved if we further tune the stepsize or the classification margin γ. However, since the focus of this study is online multiple kernel classification, we simply fix these two parameters to be 1. The above theorem also provides suggestion for the parameter β. It is important to note that the value for β suggested in the bound could be highly overestimated due to the rough approximation of $\sum_{t=1}^{T} z_i^t$ as T. We will examine empirically how β affects the prediction accuracy of the proposed algorithm.

The main shortcoming of Algorithm 1 is its high computational cost. First, at step 5, to make a prediction $\hat{y}_t$, Algorithm 1 requires combining predictions from all the kernel classifiers. Second, between step 7 and 11, Algorithm 1 requires updating all the kernel classifiers. When the number of kernels is large, both are computationally expensive. In the subsequential sections, we will study kernel selection strategies that reduce the computational cost of Algorithm 1 by selecting only a subset of kernels for prediction and updating. To distinguish from those approaches, we refer to Algorithm 1 as a deterministic approach or "OMKC$_{(\mathrm{D},\mathrm{D})}$" for short, because all the kernels are used for prediction and updating.

4 Online Multiple Kernel Classification (OMKC) Algorithms

4.1 OMKC by Stochastic Combination

Our first effort is to improve the computational efficiency of Algorithm 1 by selecting a subset of kernels for prediction. Algorithm 2 shows the key steps. It introduces the probability $q_i(t), i = 1, \ldots, m$ to denote the probability of sampling the i-th kernel at the t-th iteration, which is computed as follows:

$$q_i(t) = w_i(t) / [\max_{1 \leq j \leq m} w_j(t)] \tag{6}$$

Algorithm 2 Stochastic Combination Algorithm for OMKC (**OMKC**$_{(\mathrm{D,S})}$)

INPUT:
- Kernel functions: $k_i(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m$
- Weights $w_i(1) = 1, i = 1, \ldots, m$
- Discount weight $\beta \in (0, 1)$.
- Smoothing parameter $\delta \in (0, 1)$

for $t = 1, 2, \ldots$ **do**
  Compute $q_i(t) = w_i(t)/[\max_{1 \le j \le m} w_j(t)], i = 1, \ldots, m$
  for $i = 1, 2, \ldots, m$ **do**
    Sample $m_i(t)$ = Bernoulli_Sampling($q_i(t)$)
  end for
  Receive an instance: x_t
  Predict: $\hat{y}(t) = \mathrm{sign}\Big(\sum_{i=1}^{m} m_i(t)\mathrm{sign}\big(f_t^i(x_t)\big)\Big)$
  Receive the class label of the instance: y_t
  for $i = 1, 2, \ldots, m$ **do**
    Set $z_i(t) = 1$ if $y_t f_t^i(x_t) \le 0$ and zero otherwise
    Update $w_i(t+1) = w_i(t)\beta^{z_i(t)}$
    Update $f_{t+1}^i(x) = f_t^i(x) + z_i(t) y_t \kappa_i(x_t, x)$
  end for
end for

Only the sampled kernel classifiers will be combined to make the prediction. We refer to this stochastic selection approach as **stochastic combination**, and Algorithm 2 as **OMKC**$_{(\mathrm{D,S})}$. We have the following theorem for the mistake bound of Algorithm 2.

Theorem 2 *After receiving a sequence of T training examples, denoted by $\mathcal{L} = \{(x_t, y_t), t = 1, \ldots, T\}$, the number of mistakes M made by running Algorithm 2 is bounded as follows if $\beta = \frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$*

$$\mathrm{E}[M] \le 2\left(\left(1 + \sqrt{\frac{\ln m}{T}}\right) \min_{1 \le i \le m} F(\kappa_i, \ell, \mathcal{L}) + \ln m + \sqrt{T \ln m}\right)$$

The proof of Theorem 2 is identical to that of Theorem 1. Compared to Algorithm 1, we see that the mistake bound in expectation for Algorithm 2 remains unchanged, implying that the stochastic selection approach employed in Algorithm 2 does not affect the overall performance significantly.

4.2 OMKC by Stochastic Updating

Our second approach is to improve the learning efficiency of Algorithm 1 by sampling a subset of kernel classifiers, based on the weights assigned to kernel classifiers, for updating. Specifically, we introduce the sampling probability $p_i(t)$ which is computed by smoothing $q_i(t)$ (defined in (6)) with a uniform distribution δ/m, i.e.,

$$p_i(t) = (1 - \delta)q_i(t) + \delta/m, i = 1, \ldots, m$$

The smoothing parameter δ is introduced to guarantee that each kernel classifier will be selected with at least probability δ/m, avoiding that the sampling probability $p_i(t)$ is concentrated on a few kernels. The similar idea was also used in the study of the multi-arm

Algorithm 3 Stochastic-Update Algorithm for OMKC (**OMKC**$_{(\mathrm{S,D})}$)

INPUT:
- Kernel functions: $k_i(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m$
- Weights $w_i(1) = 1, i = 1, \ldots, m$
- Discount weight $\beta \in (0, 1)$.

for $t = 1, 2, \ldots$ **do**
  Compute $q_i(t) = w_i(t)/[\max_{1\le j\le m} w_j(t)], i = 1, \ldots, m$
  Receive an instance: x_t
  Predict: $\hat{y}(t) = \mathrm{sign}\Big(\sum_{i=1}^{m} q_i(t)\mathrm{sign}\big(f_t^i(x_t)\big)\Big)$
  Receive the class label of the instance: y_t
  Compute $p_i(t) = (1-\delta)q_i(t) + \delta/m, i = 1, \ldots, m$
  for $i = 1, 2, \ldots, m$ **do**
    Sample $m_i(t)$ = Bernoulli_Sampling($p_i(t)$)
    Set $z_i(t) = 1$ if $y_t f_t^i(x_t) \le 0$ and zero otherwise
    Update $w_i(t+1) = w_i(t)\beta^{z_i(t)m_i(t)}$
    Update $f_{t+1}^i(x) = f_t^i(x) + m_i(t)z_i(t)y_t\kappa_i(x_t, x)$
  end for
end for

bandit problem (Auer et al. 2003) to ensure the tradeoff between exploration and exploitation. Based on probabilities $p_i(t)$, we sample a subset of kernels by Bernoulli samplings which are independently conducted in each trial, one for each kernel classifier, i.e.,

$$m_i(t) = \text{Bernoulli\_Sampling}(p_i(t)), i = 1, \ldots, m$$

where $m_i(t) \in \{0, 1\}$ denotes the sampling result. The i-th kernel is selected if and only if $m_i(t) = 1$. Algorithm 3 shows the detailed steps. We refer to this kernel selection strategy as **stochastic updating** approach, and Algorithm 3 as **OMKC**$_{(\mathrm{S,D})}$. The theorem below shows the mistake bound of Algorithm 3.

Theorem 3 *After receiving a sequence of T training examples, denoted by $\mathcal{L} = \{(x_t, y_t), t = 1, \ldots, T\}$, the expected number of mistakes made by Algorithm 3, denoted by*

$$M = \mathrm{E}\left[\sum_{t=1}^{T} I\left(\sum_{i=1}^{m} q_i(t)z_i(t) \ge 0.5\right)\right],$$

is bounded as follows

$$M \le \frac{2m\ln(1/\beta)}{\delta(1-\beta)} \min_{1\le i\le m} F(\kappa_i, \ell, \mathcal{L}) + \frac{2m\ln m}{\delta(1-\beta)}$$

By choosing $\beta = \frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$, we have

$$M \le \frac{2m}{\delta}\left(\left(1 + \sqrt{\frac{\ln m}{T}}\right) \min_{1\le i\le m} F(\kappa_i, \ell, \mathcal{L}) + \ln m + \sqrt{T\ln m}\right)$$

Proof Similar to the proof for Theorems 1, we first give the lower bound and upper bound for $\ln(W_{T+1}/W_1)$ by

$$-\ln(1/\beta)\sum_{t=1}^{T} m_i(t)z_i(t) - \ln m \le \ln\left(\frac{W_{T+1}}{W_1}\right) \le -(1-\beta)\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)m_i(t)z_i(t),$$

which leads to the following inequality

$$(1-\beta)\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)m_i(t)z_i(t) \le \ln(1/\beta)\sum_{t=1}^{T} m_i(t)z_i(t) + \ln m$$

Taking expectation on both sides, we have

$$\mathrm{E}\left[(1-\beta)\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)m_i(t)z_i(t)\right] \le \mathrm{E}\left[\ln(1/\beta)\sum_{t=1}^{T} m_i(t)z_i(t)\right] + \ln m$$

Since $p_i(t) \ge \delta/m$, then

$$\frac{\delta(1-\beta)}{m}\mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)z_i(t)\right] \le \mathrm{E}\left[\ln(1/\beta)\sum_{t=1}^{T} m_i(t)z_i(t)\right] + \ln m$$

Using the result in Theorem 1, we have the following inequality for any $f \in \mathcal{H}_{\kappa_i}$

$$m_i(t)z_i(t) \le |f_i^t - f|^2_{\mathcal{H}_{\kappa_i}} - |f_i^{t+1} - f|^2_{\mathcal{H}_{k_i}} + 2\ell(f(x_t), y_t)$$

Combining the above results, we have

$$\mathrm{E}\left(\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)z_i(t)\right) \le \frac{m\ln(1/\beta)}{\delta(1-\beta)} \min_{f\in\mathcal{H}_{\kappa_i}}\left(|f|^2_{\mathcal{H}_{\kappa_i}} + 2\sum_{t=1}^{T}\ell(y_t, f(x_t))\right) + \frac{m\ln m}{\delta(1-\beta)}$$

Following the same argument as in Theorem 1, we have the result in the theorem.

As indicated in Theorem 3, the dependence of mistake bound on m is $O(m\ln m)$. Since Algorithm 3 only chooses one kernel classifier to be updated in each iteration, the algorithm is essentially similar to the multi-armed bandit problem. It is therefore not surprising to have $O(m\ln m)$ dependence for our algorithm, because the same dependence can be found in the regret bound for multi-armed bandit problem when m is the number of arms.

It is interesting to note that the mistake bound in Theorem 3 is inverse to δ, indicating that a larger δ may potentially lead to a better mistake bound for the combined kernel classifier. In the extreme case when choosing $\delta = 1$, which is equivalent to choosing the kernel classifiers uniformly at random for updating. However, in practice, we found the approach of choosing kernel classifiers uniformly at random usually leads to a poor performance because it wastes time on updating the kernel classifiers with low prediction accuracy (which could lead to poor mistake bounds due to the training on too many poor kernels). As a cautionary note about the inconsistency between the theoretical and empirical results, we conjecture that it is probably because the mistake bound is not tight enough to reveal the true behavior of the algorithm.

Besides the practical issue, another problem of choosing $\delta = 1$ is that a larger δ usually leads to a larger number of updates, as revealed by the following corollary, leading to a higher computational cost.

Corollary 1 *After receiving a sequence of T training examples, denoted by $\mathcal{L} = \{(x_t, y_t), t = 1, \ldots, T\}$, the expected number of updates made by Algorithm 3, denoted by*

$$U = \mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{m} m_i(t)z_i(t)\right],$$

Algorithm 4 Stochastic Algorithm for OMKC (**OMKC**$_{(\mathrm{S,S})}$)

INPUT:
- Kernel functions: $k_i(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m$
- Weights $w_i(1) = 1, i = 1, \ldots, m$
- Discount weight $\beta \in (0, 1)$.
- Smoothing parameter $\delta \in (0, 1)$

for $t = 1, 2, \ldots$ **do**
  Compute $q_i(t) = w_i(t)/[\max_{1\le j\le m} w_j(t)], i = 1, \ldots, m$
  Compute $p_i(t) = (1-\delta)q_i(t) + \delta/m, i = 1, \ldots, m$
  for $i = 1, 2, \ldots, m$ **do**
    Sample $m_i(t)$ = Bernoulli_Sampling($p_i(t)$)
  end for
  Receive an instance: x_t
  Predict: $\hat{y}(t) = \mathrm{sign}\Big(\sum_{i=1}^{m} m_i(t)q_i(t)\mathrm{sign}\big(f_t^i(x_t)\big)\Big)$
  Receive the class label of the instance: y_t
  for $i = 1, 2, \ldots, m$ **do**
    Set $z_i(t) = 1$ if $y_t f_t^i(x_t) \le 0$ and zero otherwise
    Update $w_i(t+1) = w_i(t)\beta^{z_i(t)m_i(t)}$
    Update $f_{t+1}^i(x) = f_t^i(x) + m_i(t)z_i(t)y_t\kappa_i(x_t, x)$
  end for
end for

is bounded as follows if $\beta = \frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$

$$U \le \frac{(1-\delta)m}{\delta}\left(\left(1+\sqrt{\frac{\ln m}{T}}\right)\min_{1\le i\le m} F(\kappa_i, \ell, \mathcal{L}) + \ln m + \sqrt{T\ln m}\right) + \delta T$$

Proof According to the definitions, we have the following result:

$$E[\sum_{t=1}^{T}\sum_{i=1}^{m} m_i(t)z_i(t)] = E[\sum_{t=1}^{T}\sum_{i=1}^{m} p_i(t)z_i(t)] \le (1-\delta)E[\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)z_i(t)] + \delta T$$

Following the same argument as in Theorem 2, we have the result in Corollary 1.

As indicated by the above corollary, a large δ usually leads to a potentially large number of updates. When δ is chosen appropriately, it could potentially improve the prediction performance through the exploration of more kernels. However, when δ is too large, it not only increases the number of updates, but also performs over-training on the poor kernels, leading to high computational cost, over-complex models, and even worse prediction accuracy.

4.3 OMKC by Stochastic Updating & Stochastic Combination

Our final approach is to combine the two kernel selection strategies, i.e., stochastic combination approach and stochastic updating approach. Algorithm 4 shows the details of this approach, to which we refer as **OMKC**$_{(\mathrm{S,S})}$. Apparently, Algorithm 4 is computationally most efficient compared to the other approaches.

4.4 Summary of Four OMKC Algorithms

By choosing different selection strategies for classifier updating and combination, we can develop several variants of OMKC algorithms. Table 1 summarizes the proposed four variants of OMKC algorithms by a mixture of different updating and combination strategies.

Table 1: Summary of the variants of OMKC Algorithms. Below, U and C denotes the selection strategies for *Update* and *Combination*, respectively; S and D denotes the *Stochastic* and *Deterministic* approaches, respectively.

Algorithm Name	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
$\mathrm{OMKC}_{(\mathrm{U,C})}$	$\mathrm{OMKC}_{(\mathrm{D,D})}$	$\mathrm{OMKC}_{(\mathrm{D,S})}$	$\mathrm{OMKC}_{(\mathrm{S,D})}$	$\mathrm{OMKC}_{(\mathrm{S,S})}$
Update strategy	Deterministic	Deterministic	Stochastic	Stochastic
Combination strategy	Deterministic	Stochastic	Deterministic	Stochastic

Among all the above four algorithms, $\mathrm{OMKC}_{(\mathrm{D,D})}$ is the most computationally intensive algorithm that updates and combines all the kernel classifiers at each iteration, while $\mathrm{OMKC}_{(\mathrm{S,S})}$ is the most efficient algorithm that selectively updates and combines a subset of kernel classifiers at each iteration. Finally, $\mathrm{OMKC}_{(\mathrm{D,S})}$ and $\mathrm{OMKC}_{(\mathrm{S,D})}$ are the other two variants of OMKC algorithms in between these two extremes. To better understand their advantages and disadvantages of these four algorithms under different situations, we will comprehensively examine their empirical performance in our experiments.

5 Experimental Results

The goal of our empirical study is to answer the following questions: (1) Whether the proposed OMKC algorithms are more effective than the regular online learning algorithms with single kernel (e.g., Perceptron) for online classification? (2) Whether the proposed OMKC algorithms are more effective than the state-of-the-art online MKL method in literature for online classification? (3) How about the efficiency and efficacy of the proposed OMKC algorithms using the stochastic strategy in comparison to the OMKC algorithms using the deterministic strategy? (4) Among all the proposed OMKC algorithms, which algorithm achieves better accuracy, efficiency, and sparsity performance? (5) How does the discount weight parameter β affects the performance of the proposed OMKC algorithms?

5.1 Experimental Testbed and Setup

In our experiments, we test the algorithms over a testbed of 15 diverse datasets [1] obtained from LIBSVM [2] and UCI machine learning repository [3]. These datasets were chosen quite arbitrarily, with different sizes and dimensions in order to examine every aspect of the performance of our algorithms. The details of these datasets are shown in Table 2.

[1] All the datasets and source code in our experiments can be downloaded from: http://www.cais.ntu.edu.sg/~chhoi/OMKC/.

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[3] http://www.ics.uci.edu/~mlearn/

Table 2: The details of 15 diverse datasets used in our experiments.

Index	Dataset	# Examples	# Dimensions	Source	Comments
D_1	ionosphere	351	34	UCI	
D_2	votes84	435	16	UCI	
D_3	wdbc	569	30	UCI	
D_4	breast	683	9	UCI	a.k.a. "wisconsin"
D_5	australian	690	14	UCI	
D_6	diabetes	768	9	UCI	a.k.a. "pima-indians"
D_7	fourclass	862	2	LIBSVM	from (Ho and Kleinberg 1996)
D_8	splice	1000	60	UCI	
D_9	dorothea	1150	100000	UCI	KDD Cup 2001
D_{10}	svmguide3	1243	22	LIBSVM	
D_{11}	svmguide1	3089	4	LIBSVM	
D_{12}	a3a	3185	123	LIBSVM	a subset of "Adult"
D_{13}	spambase	4601	57	UCI	
D_{14}	mushrooms	8124	112	UCI	
D_{15}	w5a	9888	300	LIBSVM	a subset of (Platt 1999)

We evaluate the empirical performance of the proposed online multiple kernel learning algorithms for online classification tasks. In particular, we predefine a pool of 16 kernel functions, including 3 polynomial kernels (i.e., $k(x_i, x_j) = (x_i^\top x_j)^p$) of degree parameter p=1, 2 and 3), and 13 gaussian kernels (i.e., $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/2\sigma^2)$) of kernel width parameter σ in $[2^{-6}, 2^{-5}, \ldots, 2^6]$.

We compare the proposed four variants of OMKC algorithms with the following baseline algorithms:

- **Perceptron:** the well-known Perceptron baseline algorithm with a linear kernel (Rosenblatt 1958; Freund and Schapire 1999);
- **Perceptron(u):** another Perceptron baseline algorithm with an unbiased/uniform combination of all the kernels;
- **Perceptron(*):** we conduct an online validation procedure to search for the best kernel among the pool of kernels (using the first 10% training examples), and then apply the Perceptron algorithm with the best kernel;
- **OM-2:** a state-of-the-art online learning algorithm for multiple kernel learning (Jie et al. 2010; Orabona et al. 2010);

For performance metrics, similar to the setups of a regular online learning task, we adopt the *mistake rate*, i.e., the percentage of mistakes made by the online learner over the total number of predictions made by the online learner. In addition, we also measure the size of support vectors of the classifiers learned by the online learning algorithms. Finally, we measure the average running time (including model updating and online prediction) for learning the classifiers by the online learning algorithms.

Regarding the parameter setup, for the proposed OMKC algorithms, the parameters β and δ are simply fixed to 0.8 and 0.01, respectively. We will empirically examine the parameter's impact in Section 5.6. Further, to obtain the stable average results, all online learning experiments were conducted over 20 random permutations for each dataset, and all the reported results were averaged over these 20 runs, in which every experimental run was conducted over a single pass of the permutated dataset. For the experimental environment, our experiments were evaluated on a PC with 2.3G CPU and 16GB RAM by Matlab implementation.

Table 3: Comparison of the OMKC algorithm with the OM-2 and three Perceptron based algorithms. We conducted the student t-test on the mistake results and highlighted the best results for each dataset. Among 15 datasets, OMKC$_{(D,D)}$ achieved the best on 12 datasets, while OM-2 and Perceptron(*) achieved the best on 4 datasets and 1 dataset, respectively.

Algorithm	**Perceptron**	**Perceptron (u)**	**Perceptron (*)**	**OM-2**	**OMKC$_{(D,D)}$**
Ionosphere	n= 351 d= 34 m= 16	best kernel expert: gaussian kernel of $\sigma = 1$			
Mistake (%)	26.82 ± 1.63	18.73 ± 1.23	22.07 ± 6.77	17.41 ± 1.20	**16.07 ± 1.42**
SV (#)	94.2 ± 5.7	65.8 ± 4.3	77.5 ± 23.7	128.5 ± 4.4	1547.5 ± 53.0
Time (s)	0.004 ± 0.000	0.003 ± 0.000	0.042 ± 0.002	0.113 ± 0.001	0.384 ± 0.024
votes84	n= 435 d= 16 m= 16	best kernel expert: polynomial kernel of $p = 1$			
Mistake (%)	8.17 ± 0.73	8.68 ± 0.62	9.45 ± 1.94	**7.21 ± 0.68**	**7.38 ± 0.69**
SV (#)	35.5 ± 3.2	37.8 ± 2.7	41.1 ± 8.4	56.5 ± 2.3	959.6 ± 40.2
Time (s)	0.004 ± 0.000	0.004 ± 0.000	0.045 ± 0.002	0.124 ± 0.001	0.406 ± 0.017
wdbc	n= 569 d= 30 m= 16	best kernel expert: gaussian kernel of $\sigma = 64$			
Mistake (%)	34.51 ± 1.82	41.52 ± 3.70	12.29 ± 1.01	41.70 ± 4.06	**11.70 ± 1.01**
SV (#)	196.3 ± 10.3	236.3 ± 21.0	70.0 ± 5.7	237.3 ± 23.1	3032.2 ± 37.3
Time (s)	0.007 ± 0.000	0.008 ± 0.000	0.065 ± 0.001	0.214 ± 0.007	0.597 ± 0.013
breast	n= 683 d= 9 m= 16	best kernel expert: gaussian kernel of $\sigma = 8$			
Mistake (%)	26.73 ± 1.19	41.90 ± 3.40	6.12 ± 0.79	44.33 ± 3.88	**4.86 ± 0.51**
SV (#)	182.6 ± 8.1	286.1 ± 23.2	41.8 ± 5.4	303.4 ± 26.4	1606.6 ± 49.9
Time (s)	0.008 ± 0.000	0.009 ± 0.001	0.068 ± 0.001	0.276 ± 0.015	0.641 ± 0.017
australian	n= 690 d= 14 m= 16	best kernel expert: gaussian kernel of $\sigma = 2$			
Mistake (%)	39.54 ± 1.51	39.50 ± 2.70	**38.04 ± 2.38**	39.62 ± 2.88	**37.67 ± 1.20**
SV (#)	272.9 ± 10.4	272.6 ± 18.6	262.4 ± 16.4	273.4 ± 19.9	4743.8 ± 70.0
Time (s)	0.010 ± 0.001	0.010 ± 0.001	0.091 ± 0.003	0.266 ± 0.006	0.779 ± 0.017
diabetes	n= 768 d= 8 m= 16	best kernel expert: gaussian kernel of $\sigma = 32$			
Mistake (%)	44.14 ± 1.86	45.18 ± 2.19	35.55 ± 2.07	45.35 ± 2.18	**33.69 ± 1.29**
SV (#)	339.0 ± 14.3	347.0 ± 16.8	273.0 ± 15.9	348.3 ± 16.7	4614.6 ± 63.8
Time (s)	0.012 ± 0.001	0.012 ± 0.001	0.099 ± 0.006	0.321 ± 0.014	0.886 ± 0.021
fourclass	n= 862 d= 2 m= 16	best kernel expert: gaussian kernel of $\sigma = 8$			
Mistake (%)	36.29 ± 1.09	35.82 ± 1.56	3.78 ± 0.76	35.92 ± 1.65	**3.19 ± 0.38**
SV (#)	312.8 ± 9.4	308.8 ± 13.4	32.6 ± 6.6	309.6 ± 14.2	3131.0 ± 33.5
Time (s)	0.013 ± 0.001	0.012 ± 0.001	0.092 ± 0.001	0.348 ± 0.005	0.862 ± 0.010
Splice	n= 1000 d= 60 m= 16	best kernel expert: gaussian kernel of $\sigma = 4$			
Mistake (%)	34.51 ± 1.41	30.44 ± 0.97	29.28 ± 3.84	30.79 ± 1.23	**24.57 ± 1.07**
SV (#)	345.1 ± 14.1	304.4 ± 9.7	292.8 ± 38.4	307.9 ± 12.3	5830.9 ± 90.6
Time (s)	0.015 ± 0.001	0.013 ± 0.001	0.128 ± 0.004	0.417 ± 0.013	1.122 ± 0.018
Dorothea	n= 1150 d= 100000 m= 16	best kernel expert: gaussian kernel of $\sigma = 8$			
Mistake (%)	10.07 ± 0.50	10.64 ± 0.61	11.21 ± 2.93	10.70 ± 0.72	**8.92 ± 0.37**
SV (#)	115.8 ± 5.8	122.4 ± 7.0	128.9 ± 33.7	124.1 ± 7.6	7855.3 ± 69.9
Time (s)	0.031 ± 0.001	0.031 ± 0.001	0.169 ± 0.004	0.435 ± 0.013	1.647 ± 0.071
svmguide3	n= 1243 d= 22 m= 16	best kernel expert: gaussian kernel of $\sigma = 0.5$			
Mistake (%)	32.98 ± 0.62	27.73 ± 0.85	27.18 ± 2.36	**22.84 ± 0.50**	26.00 ± 0.78
SV (#)	410.0 ± 7.7	344.7 ± 10.5	337.9 ± 29.4	812.1 ± 10.3	6107.4 ± 107.7
Time (s)	0.018 ± 0.001	0.016 ± 0.000	0.163 ± 0.004	0.695 ± 0.004	1.354 ± 0.011
svmguide1	n= 3089 d= 4 m= 16	best kernel expert: gaussian kernel of $\sigma = 32$			
Mistake (%)	23.12 ± 0.34	19.12 ± 0.56	5.68 ± 0.74	19.32 ± 0.61	**5.31 ± 0.29**
SV (#)	714.3 ± 10.4	590.5 ± 17.4	175.3 ± 22.8	596.7 ± 18.9	7089.4 ± 70.3
Time (s)	0.071 ± 0.002	0.060 ± 0.002	0.331 ± 0.007	2.116 ± 0.193	3.215 ± 0.030
a3a	n= 3185 d= 123 m= 16	best kernel expert: polynomial kernel of $p = 3$			
Mistake (%)	22.55 ± 0.51	22.09 ± 0.56	22.07 ± 0.64	**20.23 ± 0.39**	21.97 ± 0.52
SV (#)	718.1 ± 16.3	703.6 ± 17.9	702.8 ± 20.3	861.9 ± 15.8	16233.6 ± 120.3
Time (s)	0.076 ± 0.003	0.074 ± 0.002	0.502 ± 0.006	2.704 ± 0.316	4.355 ± 0.042
spambase	n= 4601 d= 57 m= 16	best kernel expert: gaussian kernel of $\sigma = 4$			
Mistake (%)	47.37 ± 0.66	58.18 ± 1.54	26.22 ± 2.09	58.16 ± 1.50	**24.36 ± 0.45**
SV (#)	2179.7 ± 30.5	2677.1 ± 71.0	1206.6 ± 96.3	2676.1 ± 69.0	27390.0 ± 180.0
Time (s)	0.321 ± 0.006	0.385 ± 0.018	0.913 ± 0.038	5.184 ± 0.239	9.653 ± 0.141
mushrooms	n= 8124 d= 112 m= 16	best kernel expert: gaussian kernel of $\sigma = 0.25$			
Mistake (%)	1.38 ± 0.10	0.51 ± 0.04	0.38 ± 0.02	0.37 ± 0.04	**0.32 ± 0.04**
SV (#)	112.2 ± 8.4	41.4 ± 3.6	30.6 ± 1.4	47.1 ± 2.8	9874.7 ± 74.1
Time (s)	0.074 ± 0.003	0.053 ± 0.002	0.777 ± 0.004	8.691 ± 0.070	9.275 ± 0.124
w5a	n= 9888 d= 300 m= 16	best kernel expert: gaussian kernel of $\sigma = 2$			
Mistake (%)	12.71 ± 0.14	3.09 ± 0.09	3.27 ± 0.31	**2.88 ± 0.10**	3.19 ± 0.10
SV (#)	1257.1 ± 13.7	305.3 ± 8.5	323.1 ± 31.1	2007.6 ± 58.6	28705.4 ± 143.4
Time (s)	0.419 ± 0.009	0.137 ± 0.004	1.284 ± 0.023	17.548 ± 2.325	25.191 ± 0.268

5.2 Evaluation of the Deterministic OMKC Algorithm

Table 3 summarizes the average experimental results for comparing the proposed OMKC$_{(\mathrm{D},\mathrm{D})}$ algorithm with three Perceptron based algorithms (i.e., Perceptron, Perceptron(u) and Perceptron(*)) and the OM-2 algorithm for online MKL, on the 15 datasets. Based on the experimental results, we performed the student t-tests and highlighted the best results in the table, including statistically no different results (w.r.t. the top 1 result) according to the student t-tests. We discuss the performance comparison as follows.

First of all, we examine the performance of the three Perceptron based algorithms. We observe that the Perceptron algorithm using the unbiased combination of all kernels usually outperforms the regular Perceptron using a linear kernel, except for a few datasets (e.g., wdbc, breast, and spambase) where Perceptron(u) is considerably worse than Perceptron with a linear kernel. Further, among all the three Perceptron algorithms, Perceptron(*) with the best kernel significantly outperforms other two algorithms for most cases, except for a couple of datasets (e.g., ionosphere and votes84). This result shows that it is important to identify the best kernel for an online learning task.

Secondly, we examine the performance of the OM-2 algorithm with comparisons to the three Perceptron algorithms. We observe that this online MKL algorithm is often more effective than or at least comparable to the two regular Perceptron algorithms, i.e., Perceptron with a linear kernel and Perceptron(u) using an unbiased combined kernel. In addition, by comparing OM-2 with Perceptron(*) that uses the best kernel, we found they are in general quite comparable, in which Perceptron(*) tends to perform considerably better on some datasets (such as australian, diabetes, wdbc, breast, fourclass, splice, svmguide1), while OM-2 tends to perform better on the other datasets. This observation shows that both identifying the best kernel and combining multiple kernels approaches are important and can exhibit their advantages for different scenarios in online learning tasks.

Thirdly, among all the compared algorithms, OMKC$_{(\mathrm{D},\mathrm{D})}$ overall achieves the best performance, which obtained the best results on 12 out of 15 datasets, significantly outperforming both Perceptron(*) and OM-2 algorithms which only obtained the best results on 1 and 3 out of 15 datasets, respectively. By further comparing the performance of the proposed OMKC$_{(\mathrm{D},\mathrm{D})}$ algorithm with Perceptron(*) in detail, we found that it consistently outperforms Perceptron(*) almost on all datasets. This shows that OMKC$_{(\mathrm{D},\mathrm{D})}$ is excellent in tracking the best kernel classifier in the online learning task. Finally, by comparing OMKC$_{(\mathrm{D},\mathrm{D})}$ with OM-2, we found that it often outperforms OM-2 for most datasets, excepts three datasets (svmguide3, a3a, w5a) where OM-2 tends to perform better. This encouraging result shows that the proposed OMKC$_{(\mathrm{D},\mathrm{D})}$ algorithm is more effective in combining multiple kernels than the state-of-the-art online MKL algorithm for online learning tasks.

Finally, despite the considerably better predictive performance achieved by the proposed OMKC$_{(\mathrm{D},\mathrm{D})}$ algorithm, we notice that it has two limitations. The first limitation is the high computational cost for learning and prediction. This is because OMKC$_{(\mathrm{D},\mathrm{D})}$ adopts the deterministic updating strategy which has to check every kernel classifier at each iteration, and whenever there is a mistake for each kernel classifier, OMKC$_{(\mathrm{D},\mathrm{D})}$ has to update the kernel classifier. The second limitation is the high model complexity, i.e., the size of support vectors learned by OMKC$_{(\mathrm{D},\mathrm{D})}$ is significantly larger than the other algorithms. For example, on dataset "ionosphere", the size of support vectors learned by OMKC$_{(\mathrm{D},\mathrm{D})}$ is almost 12 times over that by OM-2. The high learning and model complexities make OMKC$_{(\mathrm{D},\mathrm{D})}$ less efficient and attractive for some applications with a large number of kernels. This also

indicates the importance of exploring the stochastic variants of OMKC algorithms, as to be evaluated in the next section.

5.3 Evaluation of Stochastic Strategies for OMKC Algorithms

In this section, we evaluate the performance of several variants of OMKC algorithms using different stochastic strategies. In particular, we examine two kinds of stochastic strategies: stochastic updating and stochastic combinations in comparison to their deterministic counterparts. Table 4 shows the experimental results by comparing the deterministic $\text{OMKC}_{(\text{D,D})}$ algorithm with the other three OMKC algorithms. Similarly, we highlighted the best results in the table after performing student t-tests on the mistake rates. We analyze the experimental results as follows in detail.

5.3.1 Deterministic Updating v.s. Stochastic Updating

By comparing the results of different OMKC algorithms in Table 4, we have several observations for the comparisons between the deterministic and stochastic updating approaches.

First of all, by comparing $\text{OMKC}_{(\text{D,D})}$ and $\text{OMKC}_{(\text{S,D})}$ which adopt the same deterministic classifier combination approach, we found that $\text{OMKC}_{(\text{S,D})}$ using the stochastic updating strategy is able to improve both the time efficiency and model complexity over $\text{OMKC}_{(\text{D,D})}$. This is not difficult to understand since $\text{OMKC}_{(\text{S,D})}$ selectively updates a subset of kernel classifiers at each iteration, which thus runs more efficiently and produces less number of support vectors. Moreover, by examining the mistake rate results, we found that $\text{OMKC}_{(\text{S,D})}$ is able to achieve comparable or even better mistake rate than $\text{OMKC}_{(\text{D,D})}$, which validates the efficacy of the stochastic updating strategy.

Second, by comparing another pair of OMKC algorithms $\text{OMKC}_{(\text{D,S})}$ and $\text{OMKC}_{(\text{S,S})}$ which adopt the same stochastic classifier combination approach, we can draw some similar observation. In particular, $\text{OMKC}_{(\text{S,S})}$ only significantly improves the learning efficiency over $\text{OMKC}_{(\text{D,S})}$, but also achieves better or at least comparable predictive performance.

Third, by comparing the two OMKC algorithms with the stochastic updating strategy with the OM-2 algorithm, we found that both $\text{OMKC}_{(\text{S,D})}$ and $\text{OMKC}_{(\text{S,S})}$ can always achieve considerably better or at least comparable performance than the existing online MKL algorithm for most datasets.

The above observations confirm that the stochastic updating strategy can not only effectively improve the learning efficiency of OMKC, but also maintain a highly comparable or sometimes even better predictive performance for online learning tasks.

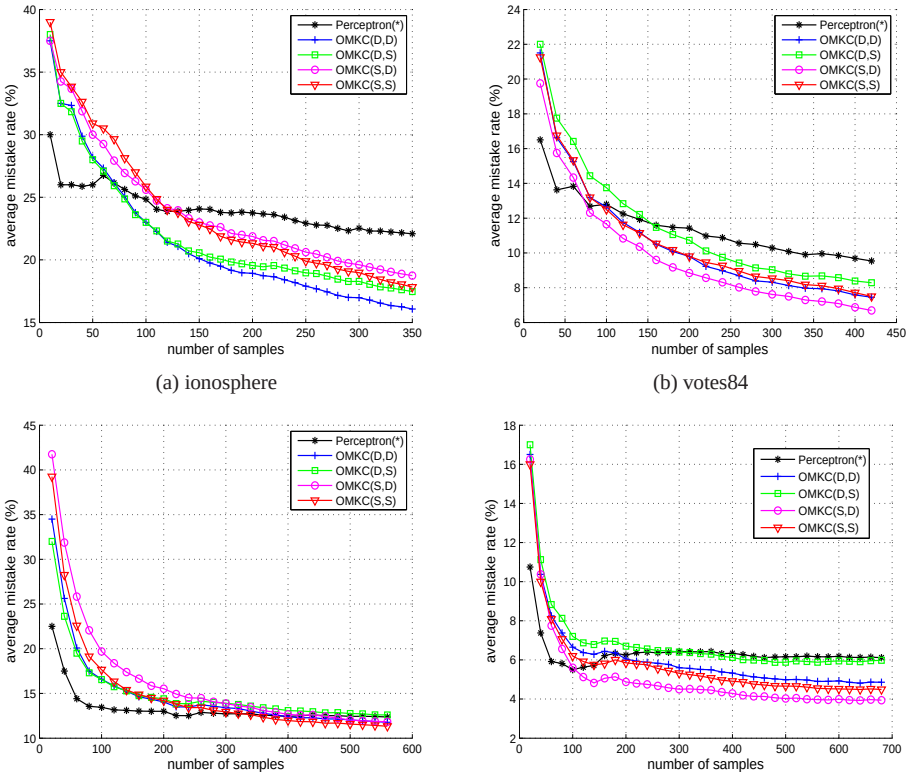
5.3.2 Deterministic Combination v.s. Stochastic Combination

Similarly, we can also draw several observations about the comparisons between the deterministic and stochastic classifier combination approaches.

First of all, by comparing $\text{OMKC}_{(\text{D,D})}$ and $\text{OMKC}_{(\text{D,S})}$ which adopt the same deterministic updating strategy, we found that $\text{OMKC}_{(\text{D,S})}$ using the stochastic classifier combination strategy is able to significantly reduce the model complexity as compared to $\text{OMKC}_{(\text{D,D})}$. For most datasets, $\text{OMKC}_{(\text{D,S})}$ is able to achieve a significant reduction of over 90 percent support vectors. Despite the significant gain in the model complexity, $\text{OMKC}_{(\text{D,S})}$ tends to result in slightly worse predictive performance; nonetheless, the mistake rate results achieved by $\text{OMKC}_{(\text{D,S})}$ remain quite competitive as compared to those by OM-2.

Second, by comparing $\text{OMKC}_{(\text{S,D})}$ and $\text{OMKC}_{(\text{S,S})}$ which adopt the same stochastic updating strategy, we found that $\text{OMKC}_{(\text{S,S})}$ using the stochastic combination strategy is able to significantly reduce both the model complexity and computational time cost. In addition, by examining the predictive performance, $\text{OMKC}_{(\text{S,S})}$ achieves fairly comparable mistake rates as compared to $\text{OMKC}_{(\text{S,D})}$.

The above observations show that the OMKC algorithms using the stochastic classifier combination strategy are able to considerably improve the learning efficacy by maintaining comparable predictive performance as compared to their counterparts.



(a) ionosphere (b) votes84

(c) wdbc (d) breast

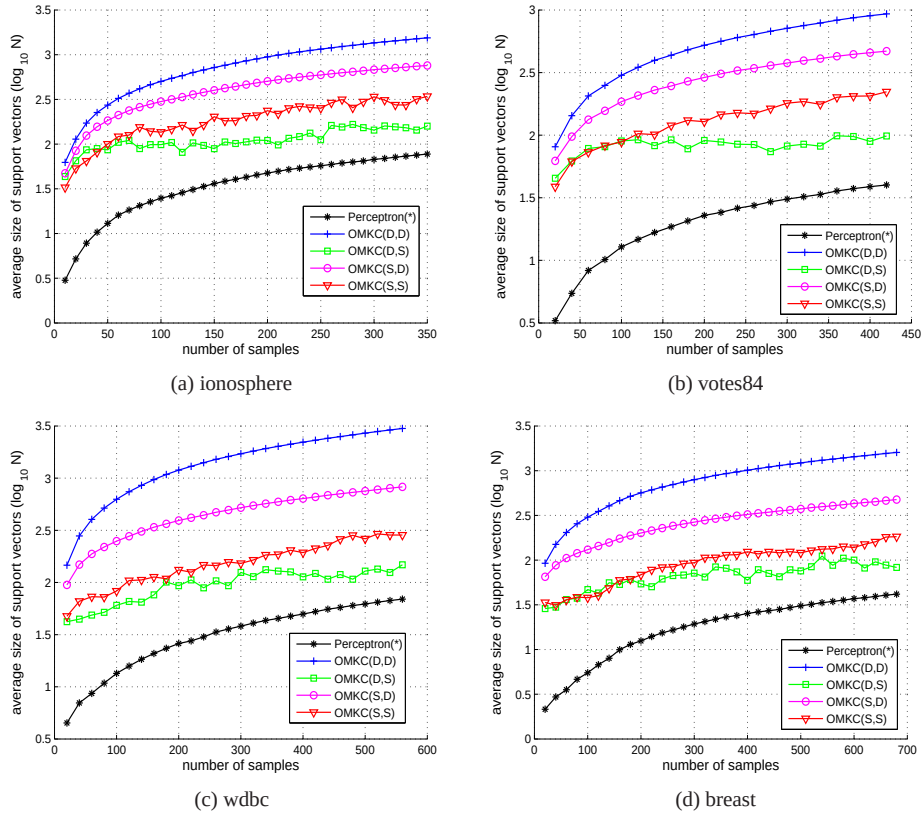
Fig. 1: Comparison of average mistake rates during the online learning processes

5.4 Evaluation of Varied Online Learning Sizes

To further examine the performance of OMKC algorithms with respect to different sizes of datasets for online learning, Figure 1 and Figure 2 show the changes of both average mistake rates and average numbers of support vectors during the online learning processes. Similar observations can be drawn from the experimental results as follows.

Table 4: Comparison of four OMKC algorithms and the OM-2 algorithm. We conducted student t-tests on the mistake rates and highlighted the best results for each dataset. **OM-2** only achieved the best for 3 out of 15 datasets, while the best **OMKC** algorithm achieved the best for 11 out of 15 datasets.

Algorithm	**OM-2**	**OMKC**$_{(\mathrm{D},\mathrm{D})}$	**OMKC**$_{(\mathrm{D},\mathrm{S})}$	**OMKC**$_{(\mathrm{S},\mathrm{D})}$	**OMKC**$_{(\mathrm{S},\mathrm{S})}$
Ionosphere	n= 351 d= 34 m= 16 best kernel expert: gaussian kernel of $\sigma = 1$				
Mistake (%)	17.41 ± 1.20	**16.07 ± 1.42**	17.21 ± 1.32	17.74 ± 1.37	17.45 ± 1.51
SV (#)	128.5 ± 4.4	1547.5 ± 53.0	184.6 ± 62.8	828.0 ± 46.2	344.6 ± 102.5
Time (s)	0.113 ± 0.001	0.384 ± 0.024	0.342 ± 0.007	0.326 ± 0.003	0.218 ± 0.006
votes84	n= 435 d= 16 m= 16 best kernel expert: polynomial kernel of $p = 1$				
Mistake (%)	**7.21 ± 0.68**	**7.38 ± 0.69**	8.02 ± 0.60	**6.71 ± 0.55**	**7.22 ± 0.99**
SV (#)	56.5 ± 2.3	959.6 ± 40.2	100.8 ± 52.6	511.4 ± 25.6	245.8 ± 76.5
Time (s)	0.124 ± 0.001	0.406 ± 0.017	0.391 ± 0.004	0.384 ± 0.003	0.252 ± 0.009
wdbc	n= 569 d= 30 m= 16 best kernel expert: gaussian kernel of $\sigma = 64$				
Mistake (%)	41.70 ± 4.06	**11.70 ± 1.01**	**11.66 ± 0.75**	12.13 ± 0.85	**11.16 ± 0.94**
SV (#)	237.3 ± 23.1	3032.2 ± 37.3	117.3 ± 54.0	918.5 ± 49.4	361.0 ± 105.0
Time (s)	0.214 ± 0.007	0.597 ± 0.013	0.588 ± 0.014	0.513 ± 0.003	0.286 ± 0.009
breast	n= 683 d= 9 m= 16 best kernel expert: gaussian kernel of $\sigma = 8$				
Mistake (%)	44.33 ± 3.88	4.86 ± 0.51	5.43 ± 0.46	**4.24 ± 0.45**	**4.36 ± 0.56**
SV (#)	303.4 ± 26.4	1606.6 ± 49.9	89.5 ± 59.7	545.9 ± 28.5	220.6 ± 74.3
Time (s)	0.276 ± 0.015	0.641 ± 0.017	0.634 ± 0.017	0.602 ± 0.007	0.350 ± 0.013
australian	n= 690 d= 14 m= 16 best kernel expert: gaussian kernel of $\sigma = 2$				
Mistake (%)	39.62 ± 2.88	**37.67 ± 1.20**	**38.13 ± 1.50**	39.64 ± 1.46	38.60 ± 1.38
SV (#)	273.4 ± 19.9	4743.8 ± 70.0	522.4 ± 272.1	3020.4 ± 82.2	1911.0 ± 418.9
Time (s)	0.266 ± 0.006	0.779 ± 0.017	0.766 ± 0.017	0.731 ± 0.006	0.538 ± 0.013
diabetes	n= 768 d= 8 m= 16 best kernel expert: gaussian kernel of $\sigma = 32$				
Mistake (%)	45.35 ± 2.18	**33.69 ± 1.29**	34.00 ± 1.41	**32.53 ± 1.30**	**33.01 ± 1.26**
SV (#)	348.3 ± 16.7	4614.6 ± 63.8	407.2 ± 223.7	3084.7 ± 81.1	2135.2 ± 410.9
Time (s)	0.321 ± 0.014	0.886 ± 0.021	0.878 ± 0.022	0.833 ± 0.018	0.626 ± 0.016
fourclass	n= 862 d= 2 m= 16 best kernel expert: gaussian kernel of $\sigma = 8$				
Mistake (%)	35.92 ± 1.65	**3.19 ± 0.38**	3.54 ± 0.42	**3.06 ± 0.37**	**3.31 ± 0.45**
SV (#)	309.6 ± 14.2	3131.0 ± 33.5	97.5 ± 33.1	600.7 ± 22.5	117.8 ± 44.3
Time (s)	0.348 ± 0.005	0.862 ± 0.010	0.850 ± 0.004	0.747 ± 0.012	0.355 ± 0.016
Splice	n= 1000 d= 60 m= 16 best kernel expert: gaussian kernel of $\sigma = 4$				
Mistake (%)	30.79 ± 1.23	**24.57 ± 1.07**	25.03 ± 0.89	26.86 ± 1.27	26.55 ± 1.15
SV (#)	307.9 ± 12.3	5830.9 ± 90.6	251.6 ± 61.1	3352.7 ± 90.2	1627.2 ± 633.8
Time (s)	0.417 ± 0.013	1.122 ± 0.018	1.086 ± 0.015	1.037 ± 0.013	0.696 ± 0.018
Dorothea	n= 1150 d= 100000 m= 16 best kernel expert: gaussian kernel of $\sigma = 8$				
Mistake (%)	10.70 ± 0.72	8.92 ± 0.37	9.92 ± 0.50	**8.03 ± 0.46**	**8.31 ± 0.46**
SV (#)	124.1 ± 7.6	7855.3 ± 69.9	230.3 ± 127.6	1492.8 ± 41.9	537.6 ± 95.8
Time (s)	0.435 ± 0.013	1.647 ± 0.071	1.597 ± 0.020	1.130 ± 0.015	0.662 ± 0.024
svmguide3	n= 1243 d= 22 m= 16 best kernel expert: gaussian kernel of $\sigma = 0.5$				
Mistake (%)	**22.84 ± 0.50**	26.00 ± 0.78	26.55 ± 0.73	**23.05 ± 0.50**	23.91 ± 0.95
SV (#)	812.1 ± 10.3	6107.4 ± 107.7	448.2 ± 164.5	3900.0 ± 68.3	2174.1 ± 573.9
Time (s)	0.695 ± 0.004	1.354 ± 0.011	1.292 ± 0.008	1.273 ± 0.008	0.902 ± 0.011
svmguide1	n= 3089 d= 4 m= 16 best kernel expert: gaussian kernel of $\sigma = 32$				
Mistake (%)	19.32 ± 0.61	5.31 ± 0.29	5.60 ± 0.31	**4.78 ± 0.25**	**4.90 ± 0.30**
SV (#)	596.7 ± 18.9	7089.4 ± 70.3	253.7 ± 222.1	2286.6 ± 104.6	970.4 ± 239.5
Time (s)	2.116 ± 0.193	3.215 ± 0.030	3.038 ± 0.021	2.770 ± 0.030	1.493 ± 0.040
a3a	n= 3185 d= 123 m= 16 best kernel expert: polynomial kernel of $p = 3$				
Mistake (%)	**20.23 ± 0.39**	21.97 ± 0.52	22.57 ± 0.52	**20.19 ± 0.42**	**20.49 ± 0.50**
SV (#)	861.9 ± 15.8	16233.6 ± 120.3	1044.4 ± 541.5	8361.4 ± 143.7	4758.7 ± 919.3
Time (s)	2.704 ± 0.316	4.355 ± 0.042	4.211 ± 0.050	3.590 ± 0.021	2.360 ± 0.031
spambase	n= 4601 d= 57 m= 16 best kernel expert: gaussian kernel of $\sigma = 4$				
Mistake (%)	58.16 ± 1.50	**24.36 ± 0.45**	24.72 ± 0.46	**24.34 ± 0.40**	**24.13 ± 0.49**
SV (#)	2676.1 ± 69.0	27390.0 ± 180.0	1169.3 ± 269.5	14316.6 ± 180.6	6324.3 ± 2070.9
Time (s)	5.184 ± 0.239	9.653 ± 0.141	9.445 ± 0.097	6.341 ± 0.072	3.832 ± 0.063
mushrooms	n= 8124 d= 112 m= 16 best kernel expert: gaussian kernel of $\sigma = 0.25$				
Mistake (%)	0.37 ± 0.04	0.32 ± 0.04	0.37 ± 0.03	**0.29 ± 0.05**	0.35 ± 0.04
SV (#)	47.1 ± 2.8	9874.7 ± 74.1	113.0 ± 37.4	694.5 ± 16.1	128.9 ± 40.2
Time (s)	8.691 ± 0.070	9.275 ± 0.124	9.031 ± 0.099	6.669 ± 0.048	2.724 ± 0.129
w5a	n= 9888 d= 300 m= 16 best kernel expert: gaussian kernel of $\sigma = 2$				
Mistake (%)	2.88 ± 0.10	3.19 ± 0.10	3.78 ± 0.10	**2.54 ± 0.07**	2.71 ± 0.10
SV (#)	2007.6 ± 58.6	28705.4 ± 143.4	424.8 ± 160.6	4403.0 ± 127.3	1568.7 ± 653.9
Time (s)	17.548 ± 2.325	25.191 ± 0.268	24.843 ± 0.253	9.449 ± 0.079	4.727 ± 0.085



(a) ionosphere (b) votes84

(c) wdbc (d) breast

Fig. 2: Comparison of average sizes of support vectors during the online learning processes

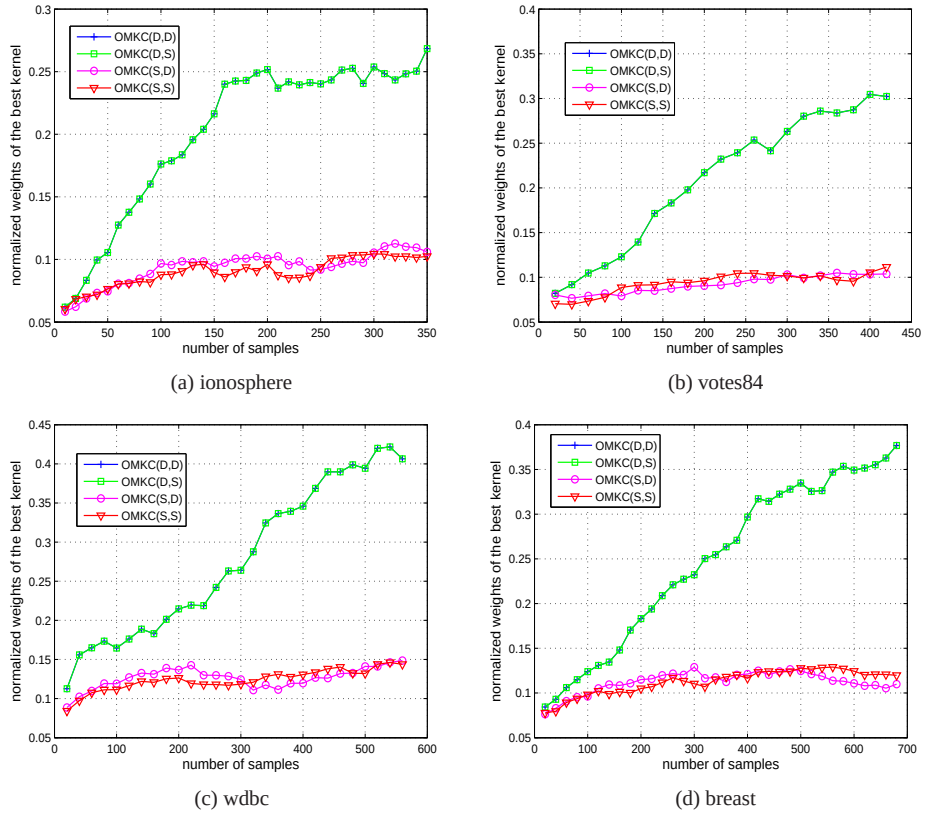
First, the results in Figure 1 validate the fact that the more the examples received in the online learning process, the better the performance achieved by the proposed OMKC algorithms. In particular, at the beginning of an online learning task, the Perceptron(*) algorithm with the best kernel is able to produce a smaller mistake rate than all of the OMKC algorithms; when more training examples are received, we found that the predictive performance of the OMKC algorithms can be improved more rapidly than the Perceptron(*) algorithm.

Second, the results in Figure 2 again verified that the $\text{OMKC}_{(\text{D,D})}$ algorithm produces the most complex classifier among all the OMKC algorithms, while $\text{OMKC}_{(\text{D,S})}$ produces the simplest classifier, which is even more sparse than $\text{OMKC}_{(\text{S,S})}$ when more training examples are received in the online learning process. This seems a little bit surprising, but it is not difficult to interpret this result. The reason is because $\text{OMKC}_{(\text{D,S})}$ using the deterministic updating strategy always increases the weights of those good kernels and meanwhile decreases the weights of those poor kernels. As a result, when applying the stochastic combination strategy, only the very small set of good kernels will be selected for the final classifier combination (most other poor kernels will be discarded due to their small sampling weights).

Third, it seems to be a little bit surprising to observe that $\text{OMKC}_{(\text{S,D})}$ sometimes can perform even better than the deterministic $\text{OMKC}_{(\text{D,D})}$ algorithm; and similarly $\text{OMKC}_{(\text{S,S})}$

usually performs better than $\text{OMKC}_{(\text{D,S})}$. However, this is possible and reasonable. We believe that it is likely to be caused by the fact that although $\text{OMKC}_{(\text{D,D})}$ is good at tracking the best kernel, it is not always guaranteed to achieve the best performance primarily because $\text{OMKC}_{(\text{D,D})}$ may rely too much on the best kernel classifier which itself may not always outperform a combination of multiple kernels; in contrast, $\text{OMKC}_{(\text{S,D})}$ is able to exploit all the kernel classifiers more effectively by the stochastic updating strategy, and thus achieves a good tradeoff between tracking the best kernel classifier and combining multiple kernel classifiers.

To further verify the above argument, Figure 3 shows the evolution results of the normalized weights in tracking the best kernel classifier by different OMKC algorithms. These weights indicate how confident the algorithm trusts on the best kernel in the online learning process. From the results, it is clear to verify that $\text{OMKC}_{(\text{D,D})}$ and $\text{OMKC}_{(\text{D,S})}$ using the deterministic updating strategy are significantly better than $\text{OMKC}_{(\text{S,D})}$ and $\text{OMKC}_{(\text{S,S})}$ for tracking the best kernel. These results are consistent to our previous analysis on the relationships between several different OMKC algorithms.



(a) ionosphere (b) votes84

(c) wdbc (d) breast

Fig. 3: Evaluation of the normalized weights in tracking the best kernel achieved by different OMKC algorithms

5.5 Evaluation of Computational Time Cost

In addition to learning accuracy, time efficiency is another important concern for online learning. In our experiments, we also examine the time efficiency by comparing different OMKC algorithms. In particular, we are interested in examining how the stochastic algorithms can reduce the computational time cost of the deterministic OMKC$_{(\mathrm{D,D})}$ algorithm. In our implementation, all the kernels are pre-computed and stored in memory, which is to facilitate the evaluation purpose. We report the average CPU running times obtained from the 20 runs that differ in the random draw of the sequential training examples.

Table 5: Evaluation of time efficiency by several different OMKC algorithms. The last three columns show the time cost ratio between each of the other three OMKC algorithms over the OMKC(D,D) algorithm.

Time (seconds)		OMKC algorithms				Time Ratio		
Dataset	Size	(D,D)	(D,S)	(S,D)	(S,S)	$\frac{(D,S)}{(D,D)}$	$\frac{(S,D)}{(D,D)}$	$\frac{(S,S)}{(D,D)}$
sonar	208	0.214	0.211	0.202	0.146	98.6 %	94.5 %	68.3 %
ionosphere	351	0.384	0.342	0.326	0.218	89.1 %	85.0 %	56.8 %
votes84	435	0.406	0.391	0.384	0.252	96.3 %	94.7 %	62.1 %
wdbc	569	0.597	0.588	0.513	0.286	98.5 %	86.0 %	47.9 %
breast	683	0.641	0.634	0.602	0.350	98.9 %	93.9 %	54.5 %
australian	690	0.779	0.766	0.731	0.538	98.4 %	93.9 %	69.1 %
diabetes	768	0.886	0.878	0.833	0.626	99.0 %	94.0 %	70.6 %
fourclass	862	0.862	0.850	0.747	0.355	98.6 %	86.6 %	41.2 %
splice	1000	1.122	1.086	1.037	0.696	96.8 %	92.4 %	62.1 %
dorothea	1150	1.647	1.597	1.130	0.662	97.0 %	68.6 %	40.2 %
svmguide3	1243	1.354	1.292	1.273	0.902	95.4 %	94.0 %	66.6 %
svmguide1	3089	3.215	3.038	2.770	1.493	94.5 %	86.1 %	46.4 %
a3a	3185	4.355	4.211	3.590	2.360	96.7 %	82.4 %	54.2 %
spambase	4601	9.653	9.445	6.341	3.832	97.8 %	65.7 %	39.7 %
mushrooms	8124	9.275	9.031	6.669	2.724	97.4 %	71.9 %	29.4 %
w5a	9888	25.191	24.843	9.449	4.727	98.6 %	37.5 %	18.8 %

First of all, from the experimental results in Table 3 and Table 4, we observe that among all OMKC algorithms, OMKC$_{(\mathrm{S,S})}$ using the stochastic updating and combination approach is the most efficient algorithm; the two deterministic updating algorithms, OMKC$_{(\mathrm{D,D})}$ and OMKC$_{(\mathrm{D,S})}$, are the least efficient algorithms; while the OMKC$_{(\mathrm{S,D})}$ algorithm is slower than OMKC$_{(\mathrm{S,S})}$, but faster than OMKC$_{(\mathrm{D,D})}$.

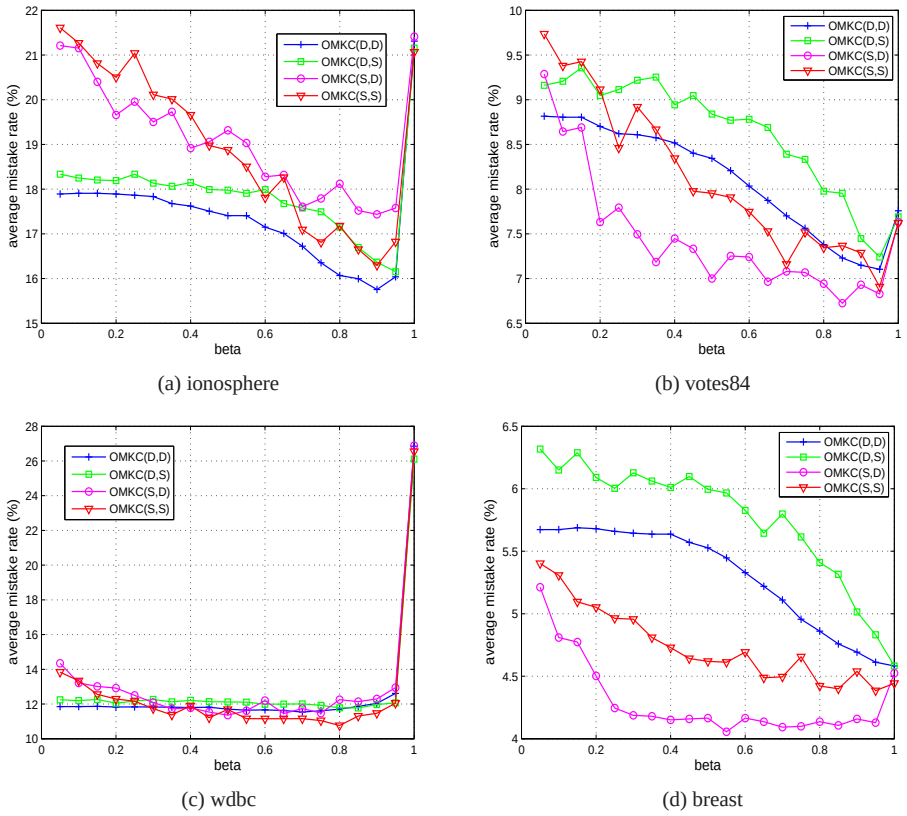
To further examine the results comprehensively, Table 5 further shows the details of quantitative evaluations of time cost and average speedup achieved by stochastic algorithms over OMKC$_{(\mathrm{D,D})}$ across various datasets. We have several observations from the results. First, similar to the previous results, we found that the two deterministic updating algorithms, OMKC$_{(\mathrm{D,D})}$ and OMKC$_{(\mathrm{D,S})}$, have similar time efficiency, in which OMKC$_{(\mathrm{D,S})}$ is slightly more efficient than OMKC$_{(\mathrm{D,D})}$ because OMKC$_{(\mathrm{D,S})}$ only selectively combines a subset of kernel classifiers during the online prediction. Second, comparing the two updating approaches for OMKC, the results again verified that algorithms using the stochastic updat-

ing are considerably more efficient than the deterministic updating algorithms. Specifically, compared with the OMKC$_{(\mathrm{D,D})}$ algorithm, OMKC$_{(\mathrm{S,S})}$ usually saves about $30\% \sim 80\%$ time cost, while OMKC$_{(\mathrm{S,D})}$ saves about $5\% \sim 60\%$ time cost.

In addition, we observe that the larger the dataset size, the more time cost typically can be saved by the two stochastic updating algorithms. For example, on dataset w5a with 9888 examples, OMKC$_{(\mathrm{S,S})}$ saves over 8% time cost, while OMKC$_{(\mathrm{S,D})}$ saves over 60% time cost. Finally, we note that the specific ratio of time cost saved by OMKC$_{(\mathrm{S,S})}$ or OMKC$_{(\mathrm{S,D})}$ is also dependent on the number of kernels. Nonetheless, the observations show that the stochastic algorithms are important and scalable for large-scale online learning tasks.

5.6 Effect of Discount Weight Parameter β

One important parameter in the proposed OMKC algorithms is the discount weight parameter β. For all the previous experiments, we simply fix parameter β to 0.8 in all situations. Although we have given the choice of β in the analysis of mistake bound, those values tend to be highly overestimated due to the approximation of $\sum_{t=1}^{T} z_i^t$ as T. In this section, we aim to empirically examine the effect of the discount weight parameter. Figure 4 shows the performance of proposed OMKC algorithms on several randomly chosen datasets with β varied from 0.05 to 1.0.



(a) ionosphere (b) votes84

(c) wdbc (d) breast

Fig. 4: Evaluation of the impact by the discount weight parameter (β).

According to Figure 4, we observe that different values of β do affect the prediction performance of the OMKC algorithms for most datasets. Further, we also observe that there is no a single best value of β that is universally optimal for every dataset. Finally, we found that for both $\text{OMKC}_{(\text{D,D})}$ and $\text{OMKC}_{(\text{S,S})}$ algorithms, the best β often falls in the range between 0.7 and 0.9. We believe that this is because when β is too small (e.g. smaller than 0.5), it penalizes too much on the misclassified kernel classifiers at the very beginning of the learning process, leading to a poor performance in finding the optimal combination weights. We note that this is in general consistent with our analysis since the optimal choice of β is $\sqrt{T}/[\sqrt{T}+\sqrt{\ln m}]$, which is a large value when T is large.

6 Discussions and Future Directions

Despite the encouraging results achieved, the current OMKC solutions can be improved in many aspects since it is a new open research problem. Below we discuss several possible directions for future research investigation.

First of all, the approach to learning each individual kernel-based classifier can be improved. The current approach used in our OMKC algorithms is an adaption of the regular kernel *Perceptron* algorithm Rosenblatt (1958); Freund and Schapire (1999). It is possible to improve the learning performance and scalability by engaging more advanced online learning algorithms (Crammer et al. 2006; Orabona et al. 2008; Zhao et al. 2012).

Second, the approach to combining the kernel-based classifiers for prediction can be improved. Instead of using the Hedge algorithm, we might explore other more general approaches that perform the online prediction with expert advices, such as the general "Follow the Perturbed Leader" approaches (Kalai and Vempala 2005; Hutter and Poland 2005).

Third, instead of assuming a finite set of given kernels, it might be possible to investigate OMKC for learning with an infinite number of kernels, which is somewhat similar to other existing infinite kernel learning studies (Argyriou et al. 2006; Chen and Ye 2008).

Fourth, the current algorithms are designed for online classification tasks. It is also possible to investigate online to batch conversion algorithms for the batch classification extension by following similar studies in literature (Dekel and Singer 2005; Dekel 2008).

Fifth, to further speedup our techniques for very large scale applications, it is possible and not difficult to parallelize our method by exploring emerging parallel computing technologies, such as multi-processor and multi-core programming techniques.

Finally, it is possible to extend the proposed OMKC framework for various real applications. For example, the current approach assumes online learning with two-class data examples. Future work is necessary to handle multi-class learning applications.

7 Conclusions

This paper investigated a new problem, "Online Multiple Kernel Classification" (OMKC), which aims to attack an online learning task by learning a kernel based prediction function from a pool of predefined kernels. To solve this challenge, we propose a novel framework by combining two types of online learning algorithms, i.e., the Perceptron algorithm that learns a classifier for a given kernel, and the Hedge algorithm that combines multiple kernel classifiers by linear weighting. The key to an OMKC task is a proper selection strategy to choose a set of kernels from the pool of predefined kernels for online classifier updates and classifier combination towards prediction. To address this key issue, we present two kinds of selection strategies: (i) deterministic approach that simply chooses all of the kernels, and (ii)

stochastic approach that randomly samples a subset of kernels according to their weights. Specifically, we proposed four variants of OMKC algorithms by adopting different online updating and combination strategies (i.e., deterministic or stochastic). It is interesting to find that each of these four OMKC algorithms enjoys different merits for different scenarios.

To examine the empirical performance of the proposed OMKC algorithms, we conducted extensive experiments on a testbed with 15 diverse real datasets. The promising results reveal three major findings: (1) all the OMKC algorithms always perform better than a regular Perceptron algorithm with an unbiased linear combination of multiple kernels, mostly perform better than the Perceptron algorithm with the best kernel found by validation, and often perform better or at least comparably than a state-of-the-art online MKL algorithm; (2) for the two different updating strategies, the stochastic updating strategy is able to significantly improve the efficiency by maintaining at least comparable performance as compared with the deterministic approach; (3) for the two different combination strategies, the deterministic combination strategy usually performs better results, while the stochastic combination strategy is able to produce a significantly more sparse classifier.

Acknowledgements

This work is in part supported by Singapore MOE Academic tier-1 grant (RG33/11), Microsoft Research grant, Office of Navy Research (ONR Award N000141210431 and N00014-09-1-0663) and National Science Foundation (IIS-0643494). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of MOE, and NSF.

Appendix A: Proof of Theorem 1

Proof The proof essentially combines the results of the Perceptron algorithm (Rosenblatt 1958) and the Hedge algorithm (Freund and Schapire 1997).

We attempt to bound $\ln(W_{T+1}/W_1)$ from both the above and the below. To achieve this goal, we first derive the upper bound of $\ln(W_{t+1}/W_t)$ as follows:

$$\ln \frac{W_{t+1}}{W_t} = \ln \left(\sum_{i=1}^{m} \frac{w_i(t)}{W_t} \beta^{z_i(t)} \right) \leq -(1-\beta) \sum_{i=1}^{m} q_i(t) z_i(t)$$

By adding the inequalities of all trials, we have

$$\ln \left(\frac{W_{T+1}}{W_1} \right) \leq -(1-\beta) \sum_{t=1}^{T} \sum_{i=1}^{m} q_i(t) z_i(t)$$

On the other hand, we have $\ln(W_{T+1}/W_1)$ lower bounded as follows

$$\ln \left(\frac{W_{T+1}}{W_1} \right) \geq \ln \frac{w_i(T+1)}{W_1} = -\ln(1/\beta) \sum_{t=1}^{T} z_i(t) - \ln m$$

Since $f_{t+1}^i(x) = f_t^i(x) + z_i(t) y_t \kappa_i(x_t, x)$, for any $f \in \mathcal{H}_{k_i}$, we have

$$\begin{aligned}
|f_{t+1}^i - f|_{\mathcal{H}_{\kappa_i}}^2 &= \langle f_t^i - f, f_t^i - f \rangle_{\mathcal{H}_{\kappa_i}} + z_i^2(t) \langle \kappa_i(x_t, \cdot), \kappa_i(x_t, \cdot) \rangle_{\mathcal{H}_{\kappa_i}} + 2 z_i(t) y_t \langle \kappa_i(x_t, \cdot), f_t^i - f \rangle_{\mathcal{H}_{\kappa_i}} \\
&= |f_t^i - f|_{\mathcal{H}_{\kappa_i}}^2 + z_i(t)^2 \kappa_i(x_t, x_t) - 2 z_i(t) y_t (f(x_t) - f_t^i(x_t)) \\
&\leq |f_t^i - f|_{\mathcal{H}_{\kappa_i}}^2 + z_i(t) + 2 z_i(t) \ell(f(x_t), y_t)) + 2 z_i(t)(-1 + y_t f_t^i(x_t)) \\
&\leq |f_t^i - f|_{\mathcal{H}_{\kappa_i}}^2 + 2 \ell(f(x_t), y_t)) - z_i(t)
\end{aligned}$$

In the last step, we use $z_i(t)y_t f_t^i(x_t) \leq 0$. We thus have

$$z_i(t) \leq -|f_{t+1}^i - f|^2_{\mathcal{H}_{k_i}} + |f_t^i - f|^2_{\mathcal{H}_{\kappa_i}} + 2\ell(f(x_t), y_t)$$

As a result, we have the following inequality hold for any $f \in \mathcal{H}_{\kappa_i}$

$$\ln\left(\frac{W_{T+1}}{W_1}\right) \geq -\ln(1/\beta)\left(|f|^2_{\mathcal{H}_{\kappa_i}} + 2\sum_{t=1}^{T}\ell(f(x_t), y_t)\right) - \ln m$$

By putting the lower and the upper bounds for $\ln(W_{T+1}/W_1)$ together, we have

$$\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)z_i(t) \leq \frac{\ln(1/\beta)}{(1-\beta)} \min_{f\in\mathcal{H}_{k_i}}\left(|f|^2_{\mathcal{H}_{\kappa_i}} + 2\sum_{t=1}^{T}\ell(f(x_t), y_t)\right) + \frac{\ln m}{1-\beta}$$

Since

$$\sum_{t=1}^{T} I\left(\sum_{i=1}^{m} q_i(t)z_i(t) \geq 0.5\right) \leq 2\sum_{t=1}^{T}\sum_{i=1}^{m} q_i(t)z_i(t),$$

we have the result in the theorem. Finally, to suggest the value for parameter β, by assuming $\sum_{t=1}^{T} z_i(t) \leq T$ and $\frac{\ln(1/\beta)}{1-\beta} \leq 1/\beta$, we can derive the solution for parameter β as follows: $\beta = \frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$, which leads to the final result as stated in the theorem.

References

Agmon, S. (1954). The relaxation method for linear inequalities, *Canadian Journal of Mathematics* **6**(3): 382–392.

Argyriou, A., Hauser, R., Micchelli, C. A. and Pontil, M. (2006). A dc-programming algorithm for kernel selection, *Proceedings of International Conference on Machine Learning*, pp. 41–48.

Auer, P., Cesa-Bianchi, N., Freund, Y. and Schapire, R. E. (2003). The nonstochastic multi-armed bandit problem, *SIAM J. Comput.* **32**(1): 48–77.

Bach, F. R., Lanckriet, G. R. G. and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm, *International Conference on Machine Learning*, ACM, New York, NY, USA, p. 6.

Bousquet, O. and Herrmann, D. J. L. (2002). On the complexity of learning the kernel matrix, *Advances in Neural Information Processing Systems*, pp. 399–406.

Cavallanti, G., Cesa-Bianchi, N. and Gentile, C. (2007). Tracking the best hyperplane with a simple budget perceptron, *Machine Learning* **69**(2-3): 143–167.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, Learning, and Games*, Cambridge University Press, New York, NY, USA.

Cesa-Bianchi, N., Mansour, Y. and Stoltz, G. (2007). Improved second-order bounds for prediction with expert advice, *Mach. Learn.* **66**(2-3): 321–352.

Chapelle, O., Weston, J. and Schölkopf, B. (2002). Cluster kernels for semi-supervised learning, *Advances in Neural Information Processing Systems*, pp. 585–592.

Chaudhuri, K., Freund, Y. and Hsu, D. (2009). A parameter-free hedging algorithm, *Advances in Neural Information Processing Systems 22*, pp. 297–305.

Chen, J. and Ye, J. (2008). Training svm with indefinite kernels, *Proceedings of the 25th International Conference on Machine learning (ICML'08)*, Helsinki, Finland, pp. 136–143.

Chen, J. and Ye, J. (2009). Multiple indefinite kernel learning with mixed norm regularization, *Proceedings of the 26th international conference on Machine learning (ICML'09)*, Montreal, Canada.

Chen, Y., Gupta, M. R. and Recht, B. (2009). Learning kernels from indefinite similarities, *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*, Montreal, Quebec, Canada, pp. 145–152.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y. (2006). Online passive-aggressive algorithms, *J. Mach. Learn. Res. (JMLR)* **7**: 551–585.

Crammer, K., Kandola, J. S. and Singer, Y. (2003). Online classification on a budget, *Advances in Neural Information Processing Systems*.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems, *J. Mach. Learn. Res. (JMLR)* **3**: 951–991.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A. and Kandola, J. S. (2001). On kernel-target alignment, *Advances in Neural Information Processing Systems*, pp. 367–373.

Dekel, O. (2008). From online to batch learning with cutoff-averaging, *NIPS*, pp. 377–384.

Dekel, O., Shalev-Shwartz, S. and Singer, Y. (2008). The forgetron: A kernel-based perceptron on a budget, *SIAM J. Comput.* **37**(5): 1342–1372.

Dekel, O. and Singer, Y. (2005). Data-driven online to batch conversions, *NIPS*.

Dredze, M. and Crammer, K. (2008). Confidence-weighted linear classification, *Proceedings of the 25th international conference on Machine learning*, ACM, pp. 264–271.

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

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm, *Machine Learning* **37**(3): 277–296.

Gentile, C. (2001). A new approximate maximal margin classification algorithm, *J. Mach. Learn. Res. (JMLR)* **2**: 213–242.

Ho, T. K. and Kleinberg, E. M. (1996). Building projectable classifiers of arbitrary complexity, *Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96)*, IEEE Computer Society, Washington, DC, USA, p. 880.

Hoi, S. C. H., Lyu, M. R. and Chang, E. Y. (2006). Learning the unified kernel machines for classification, *The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2006)*, Philadelphia, US, pp. 187–196.

Hoi, S. C., Jin, R. and Lyu, M. R. (2007). Learning non-parametric kernel matrices from pairwise constraints, *Proceedings of International Conference on Machine learning (ICML'07)*, OR, US.

Hutter, M. and Poland, J. (2005). Adaptive online prediction by following the perturbed leader, *J. Mach. Learn. Res. (JMLR)* **6**: 639–660.

Ji, S., Sun, L., Jin, R. and Ye, J. (2008). Multi-label multiple kernel learning, *Advances in Neural Information Processing Systems*.

Jie, L., Orabona, F., Fornoni, M., Caputo, B. and Cesa-bianchi, N. (2010). Om-2: An online multi-class multi-kernel learning algorithm, *Proc. of the 4th IEEE Online Learning for Computer Vision Workshop*.

Jin, R., Hoi, S. C. H. and Yang, T. (2010). Online multiple kernel learning: Algorithms and mistake bounds, *ALT*, pp. 390–404.

Jyrki Kivinen, A. J. S. and Williamson., R. C. (2004). Online learning with kernels, *IEEE Transactions on Signal Processing* **52**(8): 2165–2176.

Kalai, A. and Vempala, S. (2005). Efficient algorithms for online decision problems, *J. Comput. Syst. Sci.* **71**(3): 291–307.

Kashima, H., Tsuda, K. and Inokuchi, A. (2003). Marginalized kernels between labeled graphs, *Proceedings of International Conference on Machine Learning*, pp. 321–328.

Kivinen, J., Smola, A. J. and Williamson, R. C. (2001). Online learning with kernels, *Advances in Neural Information Processing Systems*, pp. 785–792.

Kondor, R. I. and Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces, *Proceedings of International Conference on Machine Learning*, pp. 315–322.

Kulis, B., Sustik, M. A. and Dhillon, I. S. (2009). Low-rank kernel learning with bregman matrix divergences, *J. Mach. Learn. Res.* **10**: 341–376.

Kulis, B., Sustik, M. and Dhillon, I. (2006). Learning low-rank kernel matrices, *International Conference on Machine Learning*, ACM, Pittsburgh, Pennsylvania, pp. 505–512.

Kwok, J. and Tsang, I. (2003). Learning with idealized kernels, *Proceedings of the International Conference on Machine Learning*, pp. 400–407.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E. and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming, *J. Mach. Learn. Res. (JMLR)* **5**: 27–72.

Lee, W.-J., Verzakov, S. and Duin, R. P. W. (2007). Kernel combination versus classifier combination, *7th International Workshop on Multiple Classifier Systems (MCS-2007)*, pp. 22–31.

Li, Y. and Long, P. M. (2002). The relaxed online maximum margin algorithm, *Machine Learning* **46**(1-3): 361–387.

Littlestone, N. and Warmuth, M. K. (1989). The weighted majority algorithm, *30th Annual Symposium on Foundations of Computer Science*, Research Triangle Park, North Carolina, pp. 256–261.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm, *Inf. Comput.* **108**(2): 212–261.

Martins, A. F. T., Smith, N. A., Xing, E. P., Aguiar, P. M. Q. and Figueiredo, M. A. T. (2011). Online learning of structured predictors with multiple kernels, *Journal of Machine Learning Research - Proceedings Track* **15**: 507–515.

Novikoff, A. (1962). On convergence proofs on perceptrons, *Proceedings of the Symposium on the Mathematical Theory of Automata*, Vol. XII, pp. 615–622.

Orabona, F., Keshet, J. and Caputo, B. (2008). The projectron: a bounded kernel-based perceptron, *Proceedings of International Conference on Machine Learning*, pp. 720–727.

Orabona, F., Luo, J. and Caputo, B. (2010). Online-batch strongly convex multi kernel learning, *CVPR*, pp. 787–794.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

Rakotomamonjy, A., Bach, F. R., Canu, S. and Grandvalet, Y. (2008). Simplemkl, *J. Mach. Learn. Res. (JMLR)* **11**: 2491–2521.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* **65**: 386–407.

Shalev-Shwartz, S. (2007). Online learning: Theory, algorithms, and applications, *Ph.D thesis*.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, NY, USA.

Sonnenburg, S., Rätsch, G., Schäfer, C. and Schölkopf, B. (2006). Large scale multiple kernel learning, *J. Mach. Learn. Res. (JMLR)* **7**: 1531–1565.

Tang, L., Chen, J. and Ye, J. (2009). On multiple kernel learning with multiple labels, *International Joint Conferences on Artificial Intelligence*, pp. 1255–1260.

Vovk, V. (1998). A game of prediction with expert advice, *J. Comput. Syst. Sci.* **56**(2): 153–173.

Wang, J., Zhao, P. and Hoi, S. C. H. (2012). Exact soft confidence-weighted learning, *Proceedings of the International Conference on Machine Learning*.

Xu, Z., Jin, R., King, I. and Lyu, M. R. (2008). An extended level method for efficient multiple kernel learning, *Advances in Neural Information Processing Systems (22)*.

Yang, H., Xu, Z., King, I. and Lyu, M. R. (2010). Online learning for group lasso, *The 27th International Conference on Machine Learning*, pp. 1191–1198.

Yang, T., Jin, R. and Jain, A. K. (2010). Learning from noisy side information by generalized maximum entropy model, *International Conference on Machine Learning*, pp. 1199–1206.

Zhao, P., Hoi, S. C. H. and Jin, R. (2011). Double updating online learning, *Journal of Machine Learning Research* **12**: 1587–1615.

Zhao, P., Wang, J., Wu, P., Jin, R. and Hoi, S. C. H. (2012). Fast bounded online gradient descent algorithms for scalable kernel-based online learning, *Proceedings of the International Conference on Machine Learning*.

Zhu, X., Kandola, J. S., Ghahramani, Z. and Lafferty, J. D. (2004). Nonparametric transforms of graph kernels for semi-supervised learning, *Advances in Neural Information Processing Systems*.

Zhuang, J., Tsang, I. W. and Hoi, S. C. H. (2009). Simplenpkl: simple non-parametric kernel learning, *Proceedings of International Conference on Machine Learning*, pp. 1273–1280.

Zhuang, J., Tsang, I. W. and Hoi, S. C. H. (2011a). A family of simple non-parametric kernel learning algorithms, *Journal of Machine Learning Research* **12**: 1313–1347.

Zhuang, J., Tsang, I. W. and Hoi, S. C. H. (2011b). Two-layer multiple kernel learning, *Journal of Machine Learning Research - Proceedings Track* **15**: 909–917.

Zhuang, J., Wang, J., Hoi, S. C. H. and Lan, X. (2011). Unsupervised multiple kernel learning, *Journal of Machine Learning Research - Proceedings Track* **20**: 129–144.

Zien, A. and Ong, C. S. (2007). Multiclass multiple kernel learning, *Proceedings of the 24th international conference on Machine learning (ICML'07)*, Corvalis, Oregon, pp. 1191–1198.