# Double Updating Online Learning

Peilin ZHAO
*Nanyang Technological University*

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

Rong JIN
*Michigan State University*

Journal of Machine Learning Research 12 (2011) 1587-1615 Submitted 9/09; Revised 9/10; Published 5/11

# Double Updating Online Learning

**Peilin Zhao** ZHAO0106@NTU.EDU.SG
**Steven C.H. Hoi** CHHOI@NTU.EDU.SG
*School of Computer Engineering*
*Nanyang Technological University*
*Singapore 639798*

**Rong Jin** RONGJIN@CSE.MSU.EDU
*Department of Computer Science & Engineering*
*Michigan State University*
*East Lansing, MI, 48824*

**Editor:** Nicolò Cesa-Bianchi

## Abstract

In most kernel based online learning algorithms, when an incoming instance is misclassified, it will be added into the pool of support vectors and assigned with a weight, which often remains unchanged during the rest of the learning process. This is clearly insufficient since when a new support vector is added, we generally expect the weights of the other existing support vectors to be updated in order to reflect the influence of the added support vector. In this paper, we propose a new online learning method, termed **Double Updating Online Learning**, or **DUOL** for short, that explicitly addresses this problem. Instead of only assigning a fixed weight to the misclassified example received at the current trial, the proposed online learning algorithm also tries to update the weight for one of the existing support vectors. We show that the mistake bound can be improved by the proposed online learning method. We conduct an extensive set of empirical evaluations for both binary and multi-class online learning tasks. The experimental results show that the proposed technique is considerably more effective than the state-of-the-art online learning algorithms. The source code is available to public at `http://www.cais.ntu.edu.sg/~chhoi/DUOL/`.

**Keywords:** online learning, kernel method, support vector machines, maximum margin learning, classification

## 1. Introduction

Online learning has been studied extensively in the machine learning community (Rosenblatt, 1958; Freund and Schapire, 1999; Kivinen et al., 2001; Crammer et al., 2006; Cesa-Bianchi and Lugosi, 2006). In general, for a misclassified example, most of the kernel based online learning algorithms will simply assign to it a fixed weight that remains unchanged during the whole learning process. Although such an approach is advantageous in computational efficiency, it has significant limitations. This is because when a new example is added to the pool of support vectors, the weights assigned to the existing support vectors may no longer be optimal, and should be updated to reflect the influence of the new support vector. We emphasize that although several online algorithms are proposed to update the example weights as the learning process proceeds, most of them are not designed to improve the classification accuracy. For instance, in Orabona et al. (2008) and Crammer et al. (2003); Dekel et al. (2008), online learning algorithms are proposed to adjust the example weights in order to fit in the constraint on the number of support vectors; in Kivinen et al. (2001), example weights are adjusted to deal with the drifting concepts.

Motivated by the above observations, we propose a new strategy for online learning that explicitly addresses this problem. It is designed to dynamically tune the weights of support vectors in order to improve the classification performance. In some trials of online learning, besides assigning a weight to the misclassified example, the proposed online learning algorithm also updates the weight for one of the existing support vectors, referred to as *auxiliary example*. We refer to the proposed approach as **Double Updating Online Learning** (Zhao et al., 2009), or **DUOL** for short.

The key challenge in the proposed online learning approach is to decide which existing support vector should be selected for updating weight. An intuitive choice is to select the existing support vector that "conflicts" with the new misclassified example, that is the existing support vector which on the one hand shares similar input pattern as the new example and on the other hand belongs to a class different from that of the new example. In order to quantitatively analyze the impact of updating the weight for such an existing support vector, we employ an analysis that is based on the work of online convex programming by incremental dual ascent (Shalev-Shwartz and Singer, 2006, 2007). Our analysis shows that under certain conditions, the proposed online learning algorithm can significantly reduce the mistake bound of the existing online algorithms. Besides binary classification, we extend the double updating online learning algorithm to multi-class learning. Extensive experiments show promising performance of the proposed online learning algorithm compared to the state-of-the-art algorithms for online learning.

The rest of this paper is organized as follows. Section 2 reviews the related work for online learning. Section 3 presents the proposed "double updating" approach for online learning of binary classification problems. Section 4 extends the double updating method to online multi-class learning. Section 5 gives our experimental results. Section 6 discusses the possible directions to explore in the future. Section 7 concludes this work.

## 2. Related Work

Online learning has been extensively studied in machine learning (Rosenblatt, 1958; Crammer and Singer, 2003; Cesa-Bianchi et al., 2004; Crammer et al., 2006; Fink et al., 2006). One of the most well-known online approaches is the Perceptron algorithm (Rosenblatt, 1958; Freund and Schapire, 1999), which updates the learning function by adding the misclassified example with a constant weight to the current set of support vectors. Recently a number of online learning algorithms have been developed based on the criterion of maximum margin (Crammer and Singer, 2003; Gentile, 2001; Kivinen et al., 2001; Crammer et al., 2006; Li and Long, 1999). One example is the Relaxed Online Maximum Margin algorithm (ROMMA) (Li and Long, 1999), which repeatedly chooses the hyper-planes that correctly classify the existing training examples with a large margin. Another representative example is the Passive-Aggressive (PA) algorithm (Crammer et al., 2006). It updates the classification function when a new example is misclassified or its classification score does not exceed the predefined margin. Empirical studies showed that the maximum margin based online learning algorithms are generally more effective than the Perceptron algorithm. Despite the difference, most online learningalgorithms only update the weight of the newly added support vector, and keep the weights of the existing support vectors unchanged. This constraint could significantly limit the performance of online learning.

The proposed online learning algorithm is closely related to the recent work of online convex programming by incremental dual ascent (Shalev-Shwartz and Singer, 2006, 2007). Although the idea of simultaneously updating the weights of multiple support vectors was mentioned in Shalev-Shwartz and Singer (2006, 2007), neither efficient algorithm nor theoretical result was given explicitly in their work. Besides, our work is also related to budget online learning (Weston and Bordes, 2005; Crammer et al., 2003; Cavallanti et al., 2007; Dekel et al., 2008) and online learning for drifting concepts. Although these online learning algorithms are capable of dynamically adjusting the weights of support vectors, they are designed to either fit in the budget for the number of support vectors or to handle drifting concepts, but not to reduce the number of classification mistakes in online learning.

Finally, several algorithms were proposed for online training of SVM that update the weights of more than one support vectors simultaneously (Cauwenberghs and Poggio, 2000; Bordes et al., 2005, 2007; Dredze et al., 2008; Crammer et al., 2008, 2009). In particular, in Bordes et al. (2005, 2007), the authors proposed to update the weights of two support vectors simultaneously at each iteration, similar to the proposed algorithm. These algorithms differ from the proposed one in that they are designed for efficiently learning an SVM classification model, not for online learning, and therefore do not provide guarantee for mistake bound.

## 3. Double Updating Online Learning for Binary Classification

In this section, we present the proposed double updating online learning method for solving online binary classification tasks. Below we start by introducing some preliminaries and notations.

### 3.1 Preliminaries and Notations

We consider the problem of online classification. Our goal is to learn a function $f:\mathbb{R}^d\to\mathbb{R}$ based on a sequence of training examples $\{(x_1,y_1),\ldots,(x_T,y_T)\}$, where $x_t\in\mathbb{R}^d$ is a $d$-dimensional instance and $y_t\in\mathcal{Y}=\{-1,+1\}$ is the class label assigned to $x_t$. We use $sign(f(x))$ to predict the class assignment for any $x$, and $|f(x)|$ to measure the classification confidence. Let $\ell(f(x),y):\mathbb{R}\times\mathcal{Y}\to\mathbb{R}$ be the loss function that penalizes the deviation of estimates $f(x)$ from observed labels $y$. We refer to the output $f$ of the learning algorithm as a *hypothesis* and denote the set of all possible hypotheses by $\mathcal{H}=\{f|f:\mathbb{R}^d\to\mathbb{R}\}$.

In this paper, we consider $\mathcal{H}$ a Reproducing Kernel Hilbert Space (**RKHS**) endowed with a kernel function $\kappa(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ (Vapnik, 1998) implementing the inner product$\langle\cdot,\cdot\rangle$ such that: 1) $\kappa$ has the reproducing property $\langle f,\kappa(x,\cdot)\rangle=f(x)$ for $x\in\mathbb{R}^d$; 2) $\mathcal{H}$ is the closure of the span of all $\kappa(x,\cdot)$ with $x\in\mathbb{R}^d$, that is, $\kappa(x,\cdot)\in\mathcal{H}$ for every $x\in\mathcal{X}$. The inner product $\langle\cdot,\cdot\rangle$ induces a norm on $f\in H$ in the usual way: $\|f\|_{\mathcal{H}}:=\langle f,f\rangle^{\frac{1}{2}}$. To make it clear, we use $\mathcal{H}_\kappa$ to denote an RKHS with explicit dependence on kernel function $\kappa$. Throughout the analysis, we assume $\kappa(x,x)\le 1$ for any $x\in\mathbb{R}^d$.

### 3.2 Motivation

We consider trial $t$ in an online learning task where the training example $(x_a,y_a)$ is misclassified (i.e., $y_a f(x_a)\le 0$)). Let $\mathcal{D}=\{(x_i,y_i),i=1,\ldots,n\}$ be the collection of $n$ misclassified examples received before the trial $t$. We also refer to these misclassified training examples as "support vectors". We denote by $\alpha=(\alpha_1,\ldots,\alpha_n)\in(0,C]^n$ the weights assigned to the support vectors in $\mathcal{D}$, where $C$ is a predefined constant. The resulting classifier, denoted by $f(x)$, is given by

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i).$$

In the conventional approach for online learning, we simply assign a constant weight, denoted by $\beta \in (0, C]$, to $(x_a, y_a)$, and the resulting classifier becomes

$$f'(x) = \beta y_a \kappa(x, x_a) + \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i) = \beta y_a \kappa(x, x_a) + f(x).$$

The shortcoming of the conventional online learning approach is that the introduction of the new support vector $(x_a, y_a)$ may harm the classification of existing support vectors in $\mathcal{D}$, which is revealed by the following proposition.

**Proposition 1** *Let $(x_a, y_a)$ be an example misclassified by the current classifier $f(x) = \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i)$ with $\alpha_i \geq 0, i = 1, \ldots, n$, that is, $y_a f(x_a) < 0$. Let $f'(x) = \beta y_a \kappa(x, x_a) + f(x)$ be the updated classifier with $\beta > 0$. There exists at least one support vector $x_i \in \mathcal{D}$ such that $y_i f(x_i) > y_i f'(x_i)$.*

**Proof** It follows from the fact that: $\exists\, x_i \in \mathcal{D}, y_i y_a \kappa(x_i, x_a) < 0$ when $y_a f(x_a) < 0$. ■

As indicated by Proposition 1, when a misclassified example $(x_a, y_a)$ is added to the classifier, the classification confidence of at least one existing support vector will be reduced. When $y_a f(x_a) \leq -\gamma$, there exists one support vector $(x_b, y_b) \in \mathcal{D}$ that satisfies $\beta y_a y_b k(x_a, x_b) \leq -\beta\gamma/n$. This support vector will be misclassified by the updated classifier $f'(x)$ if $y_b f(x_b) \leq \beta\gamma/n$. In order to alleviate this problem, we propose to update the weight for the existing support vector whose classification confidence is significantly affected by the new misclassified example. In particular, we consider a support vector $(x_b, y_b) \in \mathcal{D}$ for weight updating if it satisfies the following two conditions:

- $y_b f(x_b) \leq 0$, that is, support vector $(x_b, y_b)$ is misclassified by the current classifier $f(x)$;
- $k(x_b, x_a) y_a y_b \leq -\rho$ where $\rho \in (0, 1)$ is a predetermined threshold, that is, support vector $(x_b, y_b)$ "**conflicts**" with the new misclassified example $(x_a, y_a)$.

We refer to the support vector satisfying the above conditions as an **auxiliary example**. It is clear that by adding the misclassified example $(x_a, y_a)$ to classifier $f(x)$ with weight $\beta$, the classification score of $(x_b, y_b)$ will be reduced by at least $\beta\rho$, which could lead to a significant misclassification of the auxiliary example $(x_b, y_b)$. To avoid such a mistake, we propose to update the weights for both $(x_a, y_a)$ and $(x_b, y_b)$ simultaneously. In the next section, we show the details of the double updating algorithm for online learning, and the analysis for mistake bound.

Our analysis follows closely the previous work on the relationship between online learning and the dual formulation of SVM (Shalev-Shwartz and Singer, 2006, 2007), in which the online learning is interpreted as an efficient updating rule for maximizing the objective function in the dual form of SVM. We denote by $\Delta_t$ the improvement of the objective function in dual SVM when adding a misclassified example to the classification function at the $t$-th trial. According to Theorem 1 in Shalev-Shwartz and Singer (2006), if an online learning algorithm $\mathcal{A}$ is designed to ensure that for all $t$, $\Delta_t$ is bounded from below by a **bounding constant** $\Delta$, then the number of mistakes made by $\mathcal{A}$ when trained over a sequence of trials $(x_1, y_1), \ldots, (x_T, y_T)$, denoted by $M$, is upper bounded by

$$M \leq \frac{1}{\Delta}\left(\min_{f \in \mathcal{H}_\kappa} \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C\sum_{i=1}^{T} \ell(y_i f(x_i))\right),$$

where $\ell(y_i f(x_i)) = \max(0, 1 - y_i f(x_i))$ is the hinge loss function. According to Shalev-Shwartz and Singer (2006, 2007), the bounding constant $\Delta = 1/2$ when we only update the classifier with the newly misclassified example. In our analysis, we will show that $\Delta$ can be significantly improved when updating the weights for both the misclassified example and the auxiliary example.

For the remaining part of this section, we denote by $(x_b, y_b)$ an auxiliary example that satisfies the two conditions specified before. We define

$$k_a = \kappa(x_a, x_a),\ k_b = \kappa(x_b, x_b),\ k_{ab} = \kappa(x_a, x_b),\ w_{ab} = y_a y_b k_{ab}.$$

According to the assumption of auxiliary example, we have $w_{ab} = k_{ab} y_a y_b \leq -\rho$. Finally, we denote by $\widehat{\gamma}_b$ the weight for the auxiliary example $(x_b, y_b)$ that is used in the current classifier $f(x)$, by $\gamma_a$ and $\gamma_b$ the updated weights for $(x_a, y_a)$ and $(x_b, y_b)$, respectively, and by $d_{\gamma_b}$ the difference $\gamma_b - \widehat{\gamma}_b$.

## 3.3 Double Updating Online Learning for Binary Classification

Recall an auxiliary example $(x_b, y_b)$ should satisfy two conditions (I) $y_b f(x_b) \leq 0$, and (II) $w_{ab} \leq -\rho$. In addition, the example $(x_a, y_a)$ received in the current iteration $t$ is misclassified, that is, $y_a f(x_a) \leq 0$. Following the framework of dual formulation for online learning, the following lemma shows how to compute $\Delta_t$, that is, the improvement in the objective function of dual SVM by adjusting weights for $(x_a, y_a)$ and $(x_b, y_b)$.

**Lemma 1** *The maximal improvement in the objective function of dual SVM by adjusting weights for $(x_a, y_a)$ and $(x_b, y_b)$, denoted by $\Delta_t$, is computed by solving the following optimization problem(which is a special case of the optimization problem (28) in Shalev-Shwartz and Singer, 2006):*

$$\Delta_t = \max_{\gamma_a, d_{\gamma_b}} \left\{ h(\gamma_a, d_{\gamma_b}) : 0 \leq \gamma_a \leq C,\ -\widehat{\gamma}_b \leq d_{\gamma_b} \leq C - \widehat{\gamma}_b \right\} \tag{1}$$

*where*

$$h(\gamma_a, d_{\gamma_b}) = \gamma_a(1 - y_a f(x_a)) + d_{\gamma_b}(1 - y_b f(x_b)) - \frac{k_a}{2}\gamma_a^2 - \frac{k_b}{2} d_{\gamma_b}^2 - w_{ab}\gamma_a d_{\gamma_b}.$$

The lemma follows directly the dual formulation of SVM. The theorem below bounds the bounding constant $\Delta$ when $C$ is sufficiently large.

**Theorem 1** *Assume $C \geq \widehat{\gamma}_b + 1/(1-\rho)$ with $\rho \in [0, 1)$ for the selected auxiliary example $(x_b, y_b)$, we have the following bound for the bounding constant $\Delta$:*

$$\Delta \geq \frac{1}{1-\rho}.$$

**Proof** First, we show $d_{\gamma_b} \geq 0$. This is because for given $\gamma_a \geq 0$, the optimal solution for $d_{\gamma_b}$, given by

$$d_{\gamma_b} = \frac{1 - y_b f(x_b) - w_{ab}\gamma_a}{k_b},$$

is positive because $y_b f(x_b) \leq 0$ and $w_{ab} \leq -\rho$. Using the fact $k_a, k_b \leq 1$, $\gamma_a, d_{\gamma_b} \geq 0$, $y_a f(x_a) \leq 0$, $y_b f(x_b) \leq 0$, and $w_{a,b} \leq -\rho$, we have

$$h(\gamma_a, d_{\gamma_b}) \geq \gamma_a + d_{\gamma_b} - \frac{1}{2}\gamma_a^2 - \frac{1}{2}d_{\gamma_b}^2 + \rho\gamma_a d_{\gamma_b}.$$

Thus, $\Delta$ is bounded as

$$\Delta \geq \max_{\gamma_b \in [0,C], d_{\gamma_b} \in [0, C-\widehat{\gamma}_b]} \gamma_a + d_{\gamma_b} - \frac{1}{2}(\gamma_a^2 + d_{\gamma_b}^2) + \rho\gamma_a d_{\gamma_b}.$$

Under the condition that $C \geq \hat{\gamma}_b + 1/(1-\rho)$, it is easy to verify that the optimal solution for the above problem is $\gamma_a = d_{\gamma_b} = 1/(1-\rho)$, which leads to the result in the theorem. ■

We refer to the case as a **strong double update** when the condition of Theorem 1 is satisfied. We have the following theorem for the general case when we only have $C \geq 1$.

**Theorem 2** *Assume $C \geq 1$. We have the following bound for $\Delta$ when updating the weight for the misclassified example $(x_a, y_a)$ and the auxiliary example $(x_b, y_b)$:*

$$\Delta \geq \frac{1}{2} + \frac{1}{2}\min\left((1+\rho)^2, (C-\widehat{\gamma})^2\right).$$

**Proof** By setting $\gamma_a = 1$, we have $h(\gamma_a, d_{\gamma_b})$ computed as

$$h(\gamma_a = 1, d_{\gamma_b}) \geq \frac{1}{2} + (1+\rho)d_{\gamma_b} - \frac{1}{2}d_{\gamma_b}^2.$$

Hence, $\Delta$ is lower bounded by

$$\Delta \geq \frac{1}{2} + \max_{d_{\gamma_b} \in [0, C-\widehat{\gamma}]} \left((1+\rho)d_{\gamma_b} - \frac{1}{2}d_{\gamma_b}^2\right) \geq \frac{1}{2} + \frac{1}{2}\min((1+\rho)^2, (C-\widehat{\gamma})^2).$$

■

Although Theorem 1 and 2 show that the double update strategy could significantly improve the bounding constant $\Delta$ over $1/2$ and consequentially reduce the mistake bound, it is applicable only when there exists an auxiliary example. Below, we extend the double update strategy to the cases when there is no auxiliary example. Specifically, we relax the condition for performing double update as follows: there exists $(x_b, y_b) \in \mathcal{D}$ that (i) $w_{ab} \leq -\rho$, (ii) $y_b f_{t-1}(x_b) \leq 1$, and (iii) $C \geq \hat{\gamma}_b + \rho$. We refer to these cases as **weak double update**.

**Theorem 3** *Assume $w_{ab} \leq -\rho$, $y_b f_{t-1}(x_b) \leq 1$ and $C \geq \hat{\gamma}_b + \rho$, we have the following bound for the bounding constant*

$$\Delta \geq \frac{1+\rho^2}{2}.$$

**Proof** Following the definitions and assumptions, we have

$$\Delta = \max_{\gamma_a, d_{\gamma_b}} h(\gamma_a, d_{\gamma_b}) \geq h(1, \rho) \geq 1 - \frac{1}{2} + 0 - \frac{\rho^2}{2} + \rho^2 = \frac{1+\rho^2}{2}.$$

**Algorithm 1** The Double Updating Online Learning Algorithm (**DUOL**)

```
PROCEDURE
1:  Initialize $S_0 = \emptyset$, $f_0 = 0$;
2:  for t=1,2,...,T do
3:      Receive a new instance $\mathbf{x}_t$
4:      Predict $\hat{y}_t$ = sign($f_{t-1}(\mathbf{x}_t)$);
5:      Receive its label $y_t$;
6:      $l_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$
7:      if $l_t > 0$ then
8:          $w_{min} = \infty$;
9:          for $\forall i \in S_{t-1}$ do
10:             if ($f^i_{t-1} \leq 1$) then
11:                 if ($y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{min}$) then
12:                     $w_{min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
13:                     $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$;
14:                 end if
15:             end if
16:         end for
17:         $f^t_{t-1} = y_t f_{t-1}(\mathbf{x}_t)$;
18:         $S_t = S_{t-1} \cup \{t\}$;
19:         if ($w_{min} \leq -\rho$) then
20:             Compute $\gamma_t$ and $\gamma_b$ by solving
                the optimization (1)
21:             for $\forall i \in S_t$ do
22:                 $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$
                        $+ y_i(\gamma_b - \hat{\gamma}_b) y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$;
23:             end for
24:             $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$
                    $+ (\gamma_b - \hat{\gamma}_b) y_b \kappa(\mathbf{x}_b, \cdot)$;
25:         else /* no auxiliary example found */
26:             $\gamma_t = min(C, \ell_t / \kappa(x_t, x_t))$;
27:             for $\forall i \in S_t$ do
28:                 $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
29:             end for
30:             $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$;
31:         end if
32:     else
33:         $f_t = f_{t-1}$; $S_t = S_{t-1}$;
34:         for $\forall i \in S_t$ do
35:             $f^i_t \leftarrow f^i_{t-1}$;
36:         end for
37:     end if
38: end for
return $f_T$, $S_T$
END
```

Figure 1: The Algorithms of Double Updating Online Learning (DUOL).

■

Solving the optimization problem (1) is the key to the double update. The following proposition provides the optimal solution to the problem (1).

**Proposition 2** *Denote $\ell_a := 1 - y_a f(x_a)$ and $\ell_b := 1 - y_b f(x_b)$. Assume $\ell_a, \ell_b \geq 0$, $k_a, k_b > 0$ and $w_{ab} \leq 0$, then the solution of optimization problem (1) is as follows:*

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \textit{if } (k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) < 0 \textit{ and } (k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \textit{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \textit{ and } \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b) & \textit{if } \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \textit{ and } \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b \ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2}) & \textit{if } (\frac{k_b \ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2}) \in [0, C] \times [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}.$$

The detailed proof for Proposition 2 can be found in Appendix A. Figure 1 summarizes the proposed Double Updating Online Learning (DUOL) algorithm. In this algorithm, to efficiently find the auxiliary example $(x_b, y_b)$, we introduce a variable $f^i_t$ for each support vector to keep track of its classification score. Parameter $\rho$ is used to trade off between efficiency and efficacy for DUOL: the smaller $\rho$ the more double updates will be performed.

Finally, we give the mistake bound for the DUOL algorithm. We denote by $\mathcal{M}$ the set of indexes that correspond to the trials of misclassification, that is,

$$\mathcal{M} = \left\{ t \mid y_t \neq sign(f_{t-1}(x_t)), \forall t \in [T] \right\}.$$

In addition, we denote by $\mathcal{M}_d^s(\rho)$ and $\mathcal{M}_d^w(\rho)$ the sets of indexes for the cases of *strong* and *weak* double updating, respectively, that is,

$$\begin{aligned}
\mathcal{M}_d^s(\rho) &= \{t \mid \exists \text{ auxiliary example } (x_b, y_b) \text{ s.t. } C \geq \widehat{\gamma}_b + \frac{1}{1-\rho} \text{ for } (x_t, y_t),\ t \in \mathcal{M}\}, \\
\mathcal{M}_d^w(\rho) &= \{t \mid \exists\ (x_b, y_b) \text{ s.t. } w_{ab} \leq -\rho,\ y_b f_{t-1}(x_b) \leq 1 \text{ and } C \geq \widehat{\gamma}_b + \rho, t \in \mathcal{M}/\mathcal{M}_d^s(\rho)\}.
\end{aligned}$$

Note that in set $\mathcal{M}_d^s(\rho)$, for the convenience of analysis, we only consider the subset of strong updates when the condition $C \geq \widehat{\gamma}_b + 1/(1-\rho)$ is satisfied. Finally, we denote the cardinalities of sets $\mathcal{M}$, $\mathcal{M}_d^s$, and $\mathcal{M}_d^w$ by $M = |\mathcal{M}|$, $M_d^s(\rho) = |\mathcal{M}_d^s(\rho)|$, $M_d^w(\rho) = |\mathcal{M}_d^w(\rho)|$, and $M_s = M - M_d^s(\rho) - M_d^w(\rho)$, respectively.

**Theorem 4** *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of examples, where $x_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(x_t, x_t) \leq 1$ for all $t$, and assume $C \geq 1$. Then for any function $f$ in $\mathcal{H}_\kappa$, the number of prediction mistakes $M$ made by DUOL on this sequence of examples is bounded by:*

$$2\left(\min_{f \in \mathcal{H}_\kappa} \frac{1}{2}\|f\|_{\mathcal{H}_\kappa}^2 + C\sum_{i=1}^{T} \ell(y_i f(x_i))\right) - \frac{\rho^2}{2} M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

*where* $\rho \in [0, 1)$.

**Proof** According to Theorem 1 and 3, we have

$$\min_{t \in \mathcal{M}_d^s(\rho)} \Delta_t \geq \frac{1}{1-\rho},\ \min_{t \in \mathcal{M}_d^w(\rho)} \Delta_t \geq \frac{1+\rho^2}{2}.$$

Moreover, according to Theorem 2, we have $\Delta_t \geq 1/2, \forall t \in \mathcal{M}$. Putting them together, we have

$$\frac{1}{2} M_s + \frac{1+\rho^2}{2} M_d^w(\rho) + \frac{1}{1-\rho} M_d^s(\rho) \leq \left(\min_{f \in \mathcal{H}_\kappa} \frac{1}{2}\|f\|_{\mathcal{H}_\kappa}^2 + C\sum_{i=1}^{T} \ell(y_i f(x_i))\right).$$

We complete the proof using $M = M_s + M_d^w(\rho) + M_d^s(\rho)$. ■

As revealed by the above theorem, the number of mistakes made by the proposed double updating online learning algorithm will be smaller than the online learning algorithm that only performs a single update in each trial. The difference in the mistake bound is essentially due to the double updating, that is, the more the number of double updates, the more advantageous the proposed algorithm will be. Besides, the above bound also indicates that a strong double update is more powerful than a weak double update given that the associated weight of a strong double update $(1+\rho)/(1-\rho)$ is always much larger than that of a weak double update $\rho^2/2$. It is worthwhile pointing out that although according to Theorem 4, it seems that the larger the value of $\rho$ the smaller the mistake bound will be. This however may not be true because $M_d^s(\rho)$ in general decreases as $\rho$ increases. Finally, we note that Theorem 4 bounds the number of mistakes made by the proposed DUOL algorithm for $C \geq 1$. When $C < 1$, the mistake bound for the proposed algorithm follows Theorem 2, 3 and Corollary 2 in Shalev-Shwartz and Singer (2007).

## 4. Multiclass Double Updating Online Learning

In this section, we extend the proposed double updating online learning algorithm to *multiclass* learning where each instance can be assigned to multiple classes.

### 4.1 Online Multiclass Learning

Similar to online binary classification, online *multiclass* learning is performed over a sequence of training examples $(x_1, Y_1), \dots, (x_T, Y_T)$. Unlike binary classification where $y_t \in \{-1, +1\}$, in multi-class learning, each class assignment $Y_t \subseteq \mathcal{Y} = \{1, \dots, k\}$ could contain multiple class labels, making it a more challenging problem. We use $\hat{Y}_t$ to represent the class set predicted by the online learning algorithm. Before presenting our algorithm, we first review online multiclass learning (Crammer and Singer, 2003; Fink et al., 2006) based on the framework of label ranking (Crammer and Singer, 2005).

#### 4.1.1 LABEL RANKING FOR MULTICLASS LEARNING

Given an instance $x$, the label ranking approach first computes a score for every class label in $\mathcal{Y}$, and ranks the classes in the descending order of their scores. The predicted class set $\hat{Y}_t$ is formed by the classes with the highest scores. The objective of label ranking is to ensure that the score of class $r$ is significantly larger than that of class $s$ if $r \in Y_t$ is a true class assignment while $s \in \mathcal{Y} \setminus Y_t$ is not. An instance $x$ is classified incorrectly if that above condition is NOT satisfied.

We follow the protocol of *multi-prototype* (Vapnik, 1998; Crammer and Singer, 2001; Crammer et al., 2006) for the design of multiclass multilabel learning algorithm. It learns multiple hypotheses/classifiers, one classifier for each class in $\mathcal{Y}$, leading to a total of $k$ classifiers that are trained for the classification task. Specifically, for trial $t$, upon receiving an instance $x_t$, the scores of $k$ classes output by the set of $k$ hypotheses are given by

$$\overline{f}_{t-1}(x_t) = (f_{t-1,1}(x_t), \cdots, f_{t-1,k}(x_t))^T,$$

where $f_{t-1,i} \in \mathcal{H}_K, i = 1, \dots, k$. We introduce two variables $r_t$ and $s_t$ that are defined as follows:

$$r_t = \arg\min_{r \in Y_t} f_{t-1,r}(x_t) \quad \text{and} \quad s_t = \arg\max_{s \notin Y_t} f_{t-1,s}(x_t), \tag{2}$$

here, $r_t$ and $s_t$ represent the class of the smallest score among all relevant classes and the class of the largest score among the irrelevant classes, respectively. Using the notation of $r_t$ and $s_t$, the *margin* with respect to the hypothesis set $\overline{f}_{t-1}$ at trial $t$ is defined as follows:

$$\Gamma\big(\overline{f}_{t-1}; (x_t, Y_t)\big) = f_{t-1,r_t}(x_t) - f_{t-1,s_t}(x_t).$$

Based on the notation of classification margin, we define the loss function of hypotheses $\overline{f}_{t-1}(x)$ for training example $(x_t, Y_t)$ as follows:

$$\ell\big(\overline{f}_{t-1}; (x_t, Y_t)\big) = \max_{r \in Y_t, s \notin Y_t} \big[1 - (f_{t-1,r}(x_t) - f_{t-1,s}(x_t))\big]_+,$$

where $[x]_+ = \max(0, x)$.

#### 4.1.2 A Perceptron Algorithm for Online Multiclass Learning

According to Crammer and Singer (2003), when an example is misclassified at trial $t$, we update each component of the classifier $\bar{f}_{t-1}$ as follows:

$$f_{t,i}(x) = f_{t-1,i}(x) + \sigma_{Y_t}(i,t)\gamma_t \kappa(x_t, x), \ \forall i \in \mathcal{Y}, \tag{3}$$

where $\gamma_t \in (0, C]$, and function $\sigma_{Y_t}(i,t)$, which is simplified as $\sigma(i,t)$, is defined below:

$$\sigma(i,t) = \begin{cases} 1 & \text{if } i = r_t \\ -1 & \text{if } i = s_t \\ 0 & \text{otherwise} \end{cases}.$$

Using notation $H(Y_t) = \big(\sigma(1,t), \cdots, \sigma(k,t)\big)^T$, we rewrite Equation (3) as $\bar{f}_t(x) = \bar{f}_{t-1}(x) + \gamma_t H(Y_t)\kappa(x_t, x)$, or equivalently

$$\bar{f}(x) = \sum_{i=1}^{n} \gamma_i H(Y_i)\kappa(x_i, x),$$

where $n$ is the number of support vectors received so far.

### 4.2 Multiclass DUOL Algorithm

We extend the DUOL algorithm to multiclass learning. We denote by $(x_a, Y_a)$ the misclassified example received at the current trial, that is, $(\bar{f}(x_a))_{r_a} - (\bar{f}(x_a))_{s_a} \leq 0$. Similar to DUOL for binary classification, we introduce an auxiliary example $(x_b, Y_b)$ from the existing support vectors that obey the following conditions:

1. $(\bar{f}(x_b))_{r_b} - (\bar{f}(x_b))_{s_b} \leq 0$, that is, $(x_b, Y_b)$ is misclassified by current classifier $\bar{f}$;
2. $(H(Y_a) \cdot H(Y_b))\kappa(x_a, x_b) \leq -2\rho$ where $\rho \in (0,1)$ is a threshold. This property indicates that example $(x_a, Y_a)$ **conflicts** with example $(x_b, Y_b)$.

Compared to auxiliary example defined for binary classification, we introduce $H(Y_a) \cdot H(Y_b)$ in above when defining two conflicting instances. Given $\kappa(x_a, x_b) \geq 0$, the second condition of auxiliary example implies $H(Y_a) \cdot H(Y_b) \leq 0$, which further indicates that two examples $(x_a, Y_a)$ and $(x_b, Y_b)$ have the opposite prediction, that is, ($r_a = s_b$) or ($s_a = r_b$). This result is revealed by the following proposition.

**Proposition 3** *The inequality $H(Y_a) \cdot H(Y_b) < 0$ holds if and only if ($r_a = s_b$) or ($s_a = r_b$).*

The proof of Proposition 3 is given in the appendix.

Similar to the DUOL algorithm for binary classification, our analysis aims to show that by updating weights for both misclassified example and the auxiliary example, we may be able to significantly improve the bounding constant $\Delta$, which is defined as follows:

$$M \times \Delta \leq \Big( \min_{\bar{f} \in \bar{\mathcal{H}}_\kappa} F(\bar{f}) + C \sum_{i=1}^{T} \ell\big(\bar{f}; (x_i, Y_i)\big)\Big), \tag{4}$$

where $\bar{\mathcal{H}}_\kappa = \prod_{i=1}^{k} \mathcal{H}_\kappa$ and $F(\bar{f}) = \sum_{i=1}^{k} \frac{1}{2}\|f_i\|^2_{\mathcal{H}_\kappa}$. To ease our further discussions, we define $k_a = \kappa(x_a, x_a), k_b = \kappa(x_b, x_b), w_{ab} = (H(Y_a) \cdot H(Y_b))\kappa(x_a, x_b)$ .

The following proposition shows the optimization problem related to the multiclass double updating online learning algorithm, which forms the basis for deriving the bounding constant $\Delta$.

**Proposition 4** *With the double updating, that is, adjusting the weight of some auxiliary support vector $(x_b, Y_b)$ from $\hat{\gamma}_b$ to $\gamma_b$ (denoted by $d_{\gamma_b} = \gamma_b - \hat{\gamma}_b$) and assigning weight $\gamma_a$ to the current misclassified example $(x_a, Y_a)$, the improvement in the objective function of dual SVM, denoted by $\Delta_t$, is computed by the following optimization problem:*

$$\begin{aligned}\max_{\gamma_a, d_{\gamma_b}} \quad & \gamma_a\Big(1 - \big(f_{t-1,r_a}(x_a) - f_{t-1,s_a}(x_a)\big)\Big) + d_{\gamma_b}\Big(1 - \big(f_{t-1,r_b}(x_b) - f_{t-1,s_b}(x_b)\big)\Big) \\ & -k_a\gamma_a^2 - k_b d_{\gamma_b}^2 - w_{ab}\gamma_a d_{\gamma_b}, \\ s.t. \quad & 0 \le \gamma_a \le C, -\hat{\gamma}_b \le d_{\gamma_b} \le C - \hat{\gamma}_b.\end{aligned} \tag{5}$$

**Theorem 5** *Assume $\kappa(x,x) \le 1$ for any $x$ and $C \ge \hat{\gamma}_b + \frac{1}{2(1-\rho)}$ for the selected auxiliary example $(x_b, Y_b)$, we have the following bound for $\Delta$:*

$$\Delta \ge \frac{1}{2(1-\rho)}.$$

We refer to the case as a strong double update when there exists a auxiliary example $(x_b, Y_b)$ s.t. $C \ge \hat{\gamma}_b + \frac{1}{2(1-\rho)}$. Similar to double updating for binary classification, we introduce *weak* double update when there exists $(x_b, Y_b)$ s.t. $w_{ab} \le -2\rho$, $f_{t-1,r_b}(x_b) - f_{t-1,s_b}(x_b) \le 1$, and $C \ge \hat{\gamma}_b + \frac{\rho}{2}$.

**Theorem 6** *Assume there exists $(x_b, Y_b)$ s.t. $w_{ab} \le -2\rho$, $f_{t-1,r_b}(x_b) - f_{t-1,s_b}(x_b) \le 1$, $C \ge \hat{\gamma}_b + \frac{\rho}{2}$ and the current instance is misclassified, then we have the following bounding constant*

$$\Delta \ge \frac{1+\rho^2}{4}.$$

The exact solution to the Quadratic Programming (QP) problem in (5) is given by the following proposition.

**Proposition 5** *Denote $\ell_a := 1 - (f_{t-1,r_a}(x_a) - f_{t-1,s_a}(x_a))$ and $\ell_b := 1 - (f_{t-1,r_b}(x_b) - f_{t-1,s_b}(x_b))$. Assume $\ell_a, \ell_b \ge 0$, $k_a, k_b > 0$ and $w_{ab} \le 0$, then the solution of optimization (5) is as follows:*

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C-\hat{\gamma}_b) & \text{if } (2k_aC + w_{ab}(C-\hat{\gamma}_b) - \ell_a) < 0 \text{ and } (2k_b(C-\hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{2k_b}) & \text{if } \frac{w_{ab}^2C - w_{ab}\ell_b - 4k_ak_bC + 2k_b\ell_a}{2k_b} > 0 \text{ and } \frac{\ell_b - w_{ab}C}{2k_b} \in [-\hat{\gamma}_b, C-\hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{2k_a}, C-\hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{2k_a} \in [0,C] \text{ and } \ell_b - 2k_b(C-\hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{2k_a} > 0 \\ (\frac{2k_b\ell_a - w_{ab}\ell_b}{4k_ak_b - w_{ab}^2}, \frac{2k_a\ell_b - w_{ab}\ell_a}{4k_ak_b - w_{ab}^2}) & \text{if } (\frac{2k_b\ell_a - w_{ab}\ell_b}{4k_ak_b - w_{ab}^2}, \frac{2k_a\ell_b - w_{ab}\ell_a}{4k_ak_b - w_{ab}^2}) \in [0,C] \times [-\hat{\gamma}_b, C-\hat{\gamma}_b] \end{cases}.$$

We skip the proof due to its high similarity to that of Proposition 2. Figure 2 summarizes the steps of the multiclass DUOL (M-DUOL) algorithm. Note that we replace the conditions for auxiliary example with the margin error in order to make more double updates.

A mistake bound for the M-DUOL algorithm, similar to Theorem 4, is given by the following theorem.

**Theorem 7** *Let $(x_1, Y_1), \ldots, (x_T, Y_T)$ be a sequence of examples, where $x_t \in \mathbb{R}^n$, $Y_t \subseteq \mathcal{Y}$ and $\kappa(x_i, x_j) \in [0,1]$ for all $i, j$. And assume $C \ge 1$. Then for any function $\overline{f} \in \prod_{i=1}^k \mathcal{H}_\kappa$, the number of prediction mistakes $M$ made by M-DUOL on this sequence of examples is bounded by:*

$$4\Big(\min_{\overline{f} \in \mathcal{H}_\kappa} F(\overline{f}) + C\sum_{i=1}^T \ell\big(\overline{f}; (x_i, Y_i)\big)\Big) - \frac{\rho^2}{2}M_d^w(\rho) - \frac{1+\rho}{1-\rho}M_d^s(\rho).$$

**Algorithm 2:** The Multiclass DUOL Algorithm (**M-DUOL**)

```
PROCEDURE
1:  Initialize $H_0 = \emptyset$, $S_0 = \emptyset$, $\bar{f}_0 = 0$;
2:  for t=1,2,...,T do
3:      Receive a new instance $\mathbf{x}_t$
4:      Predict $W_{t-1} = \bar{f}_{t-1}(\mathbf{x}_t)$;
5:      Receive its label set $Y_t$
6:      $\ell_t = \left[1 - W_{t-1} \cdot H(Y_t)\right]_+$
7:      if $l_t > 0$ then
8:          $w_{min} = \infty$;
9:          for $\forall i \in S_{t-1}$ do
10:             if $f^i_{t-1} \leq 1$ then
11:                 if $(Hk_{ti} < w_{min}$ then
12:                     $w_{min} = Hk_{ti}$;
13:                     $(\mathbf{x}_b, Y_b) = (\mathbf{x}_i, Y_i)$;
14:                 end if
15:             end if
16:         end for
17:         $f^t_{t-1} = W_{t-1} \cdot H(Y_t)$;
18:         $S_t = S_{t-1} \cup \{t\}$; $H_t = H_{t-1} \cup \{H(Y_t)\}$;
19:         if $(w_{min} \leq -2\rho)$ then
20:             Compute $\gamma_t$ and $\gamma_b$ by solving
                 the optimization (5)
21:             for $\forall i \in S_t$ do
22:                 $f^i_t \leftarrow f^i_{t-1} + [\gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \mathbf{x}_i)] \cdot H(Y_i)$
                      $+ [(\gamma_b - \hat{\gamma}_b) * H(Y_b) * \kappa(\mathbf{x}_b, \mathbf{x}_i)] \cdot H(Y_i)$;
23:             end for
24:             $\bar{f}_t = \bar{f}_{t-1} + \gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \cdot)$
                      $+ (\gamma_b - \hat{\gamma}_b) * H(Y_b) * \kappa(\mathbf{x}_b, \cdot)$;
25:         else /* no auxiliary example found */
26:             $\gamma_t = min(C, \frac{\ell_t}{2\kappa(x_t, x_t)})$;
27:             for $\forall i \in S_t$ do
28:                 $f^i_t \leftarrow f^i_{t-1} + [\gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \mathbf{x}_i)] \cdot H(Y_i)$;
29:             end for
30:             $\bar{f}_t = \bar{f}_{t-1} + \gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \cdot)$;
31:         end if
32:     else
33:         $\bar{f}_t = \bar{f}_{t-1}$; $S_t = S_{t-1}$; $H_t = H_{t-1}$;
34:         for $\forall i \in S_t$ do
35:             $f^i_t \leftarrow f^i_{t-1}$;
36:         end for
37:     end if
38: end for
return $\bar{f}_T$, $S_T$, $H_T$
END
```

Figure 2: Algorithms of multiclass double-updating online learning (M-DUOL).

## 5. Experimental Results

In this section, we evaluate the empirical performance of the proposed double updating online learning algorithms for online learning tasks. We first evaluate the performance of DUOL for binary classification, followed by the evaluation of multiclass double updating online learning.

### 5.1 Testbeds and Experimental Setup for Binary-class Online Learning

We compare our technique with a number of state-of-the-art techniques, including the kernel Perceptron algorithm (Kivinen et al., 2001), the "ROMMA" algorithm and its aggressive version "agg-ROMMA" (Li and Long, 1999), the ALMA$_p(\alpha)$ algorithm (Gentile, 2001), and the Passive-Aggressive algorithms ("PA") (Crammer et al., 2006). For PA, two versions of algorithms (PA-I and PA-II) are implemented as described in Crammer et al. (2006). Note that one may also compare with the online SVM algorithm (Shalev-Shwartz and Singer, 2006), which updates the weights for all support vectors in each trial. However, we do not include this baseline for comparison because it is too computationally intensive to run on some large data sets.

For the proposed DUOL algorithms, we implement three variants based on different solvers to the problem in (1): (i) "DUOL$_{appr}$" that employs an approximate solution to (1), that is, $\gamma_t = \frac{1}{1-\rho}$ and $\gamma_b = \hat{\gamma}_b + \frac{1}{1-\rho}$, (ii)"DUOL" that uses the exact solution to (1) given in Proposition 2, and (iii) "DUOL$_{iter}$" that first updates the weight for the misclassified example and then the weight for auxiliary example, as suggested in Shalev-Shwartz and Singer (2007)

We test all the algorithms on eight benchmark data sets from web machine learning repositories, which are listed in table 1. All of the data sets can be downloaded from LIBSVM website,[1] UCI machine learning repository,[2] and MIT CBCL face data sets.[3]

| Data Set | # examples | # features |
|---|---:|---:|
| sonar | 208 | 60 |
| splice | 1,000 | 60 |
| german | 1,000 | 24 |
| mushrooms | 8,124 | 112 |
| dorothea | 1,150 | 100,000 |
| spambase | 4,601 | 57 |
| MITFace | 6,977 | 361 |
| w7a | 24,692 | 300 |

Table 1: Binary-class data sets used in the experiments.

To make a fair comparison, for all algorithms in comparison, we set $C = 5$ and use the same Gaussian kernel with $\sigma = 8$. For the $\text{ALMA}_p(\alpha)$ algorithm, parameter $p$ and $\alpha$ are set to 2 and 0.9, respectively, based on our experience. For the proposed DUOL algorithm, we fix $\rho$ to be 0 for all cases. All the experiments are repeated 20 times, each with an independent random permutation of the data points. All the results are reported by averaging over the 20 runs. We evaluate the online learning performance by measuring the *mistake rate*, that is, the percentage of examples that are misclassified by the online learning algorithm. We measure the sparsity of the learned classifiers by the number of support vectors. We evaluate computational efficiencies of all the algorithms in terms of their CPU running time (in seconds). All the experiments are run in Matlab over a windows machine of 2.3GHz CPU.

### 5.2 Performance Evaluation for Binary-Class Online Learning

Table 2 summarizes the performance of all the compared online learning algorithms over the binary data sets. We can draw several observations from the results.

First, among the six baseline algorithms in comparison, we observe that the agg-ROMMA and two PA algorithms (PA-I and PA-II) perform considerably better than the other three algorithms (i.e., Perceptron, ROMMA, and ALMA) in most cases. We also notice that the agg-ROMMA and the two PA algorithms consume considerably larger numbers of support vectors than the other three algorithms. We believe this is because the agg-ROMMA and the two PA algorithms adopt more aggressive strategies than the other three algorithms, resulting in more updates and better classification performance. For the convenience of discussion, we refer to agg-ROMMA and two PA algorithms as *aggressive* algorithms, and the other three online learning algorithms as *non-aggressive* ones.

Second, we observe that among the three variants of double updating online learning, the DUOL approach, which solves the optimization problem exactly, yields the least *mistake rate* with the smallest number of support vectors for most of the cases. Comparing with the baseline algorithms,

1. LIBSVM website is `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.
2. UCI ML repository is at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.
3. MIT CBCL face data sets can be found at `http://cbcl.mit.edu/software-datasets`.

| Algorithm | sonar | | | | splice | | |
|---|---|---|---|---|---|---|---|
| | Mistake (%) | Support Vectors (#) | Time (s) | | Mistakes (%) | Support Vectors (#) | Time (s) |
| Perceptron | 38.125 ± 3.815 | 79.30 ± 7.93 | **0.004** | | 27.120 ± 0.975 | 271.20 ± 9.75 | **0.017** |
| ROMMA | 36.587 ± 2.976 | **76.10 ± 6.19** | 0.006 | | 25.560 ± 0.814 | **255.60 ± 8.14** | 0.032 |
| agg-ROMMA | 34.928 ± 2.860 | 130.05 ± 7.51 | 0.009 | | 22.980 ± 0.780 | 602.90 ± 7.42 | 0.044 |
| ALMA$_2$(0.9) | 36.370 ± 3.572 | 86.25 ± 6.43 | 0.006 | | 26.040 ± 0.965 | 314.95 ± 9.41 | 0.032 |
| PA-I | 40.986 ± 2.837 | 154.15 ± 6.95 | **0.004** | | 23.815 ± 1.042 | 665.60 ± 5.60 | 0.029 |
| PA-II | 40.481 ± 3.023 | 162.40 ± 6.26 | **0.004** | | 23.515 ± 1.005 | 689.00 ± 7.85 | 0.029 |
| DUOL$_{iter}$ | 39.495 ± 3.299 | 149.85 ± 3.42 | 0.014 | | 23.205 ± 0.932 | 566.85 ±13.08 | 0.097 |
| DUOL$_{appr}$ | 41.010 ± 2.335 | 162.25 ± 5.01 | 0.013 | | 21.945 ± 1.134 | 721.85 ± 9.10 | 0.095 |
| DUOL | **34.255 ± 2.811** | 137.60 ± 6.99 | 0.017 | | **20.875 ± 0.868** | 577.15 ±10.81 | 0.087 |

| Algorithm | german | | | | mushrooms | | |
|---|---|---|---|---|---|---|---|
| | Mistake (%) | Support Vectors (#) | Time (s) | | Mistakes (%) | Support Vectors (#) | Time (s) |
| Perceptron | 34.760 ± 0.947 | 347.60 ± 9.47 | **0.019** | | 2.083 ± 0.278 | **169.25 ± 22.58** | **0.148** |
| ROMMA | 34.725 ± 1.009 | **347.25 ± 10.09** | 0.037 | | 2.429 ± 0.101 | 197.35 ± 8.24 | 0.264 |
| agg-ROMMA | 32.925 ± 1.184 | 633.40 ± 14.02 | 0.049 | | 1.568 ± 0.096 | 1307.90 ± 39.59 | 0.576 |
| ALMA$_2$(0.9) | 33.480 ± 0.681 | 394.75 ± 9.24 | 0.036 | | 2.538 ± 0.297 | 304.80 ± 38.02 | 0.267 |
| PA-I | 33.010 ± 1.025 | 721.10 ± 12.99 | 0.031 | | 1.661 ± 0.089 | 1221.55 ± 22.80 | 0.454 |
| PA-II | 32.630 ± 1.016 | 749.50 ± 11.84 | 0.032 | | 1.657 ± 0.088 | 1326.20 ± 22.85 | 0.483 |
| DUOL$_{iter}$ | 35.985 ± 1.077 | 714.35 ± 12.75 | 0.125 | | 1.537 ± 0.101 | 860.05 ± 23.00 | 0.521 |
| DUOL$_{appr}$ | **30.275 ± 0.937** | 716.10 ± 10.44 | 0.096 | | 1.459 ± 0.101 | 1291.35 ± 32.03 | 0.658 |
| DUOL | 31.810 ± 1.090 | 656.30 ± 14.36 | 0.108 | | **0.596 ± 0.053** | 453.70 ± 19.40 | 0.341 |

| Algorithm | dorothea | | | | spambase | | |
|---|---|---|---|---|---|---|---|
| | Mistake (%) | Support Vectors (#) | Time (s) | | Mistakes (%) | Support Vectors (#) | Time (s) |
| Perceptron | 13.257 ± 0.973 | **152.45 ± 11.18** | **0.016** | | 24.987 ± 0.525 | 1149.65 ± 24.16 | **0.215** |
| ROMMA | 17.461 ± 0.537 | 200.80 ± 6.18 | 0.035 | | 23.953 ± 0.510 | **1102.10 ± 23.44** | 0.275 |
| agg-ROMMA | 17.435 ± 0.500 | 438.30 ± 13.83 | 0.044 | | 21.242 ± 0.384 | 2550.70 ± 27.28 | 0.515 |
| ALMA$_2$(0.9) | 14.478 ± 0.378 | 210.25 ± 5.68 | 0.035 | | 23.579 ± 0.411 | 1550.15 ± 15.65 | 0.348 |
| PA-I | 17.500 ± 0.491 | 461.30 ± 15.80 | 0.026 | | 22.112 ± 0.374 | 2861.50 ± 24.36 | 0.479 |
| PA-II | 17.500 ± 0.491 | 461.30 ± 15.80 | 0.027 | | 21.907 ± 0.340 | 3029.10 ± 24.69 | 0.504 |
| DUOL$_{iter}$ | 21.109 ± 0.796 | 559.20 ± 19.44 | 0.080 | | 21.907 ± 0.432 | 2511.20 ± 34.14 | 1.215 |
| DUOL$_{appr}$ | 17.500 ± 0.491 | 461.30 ± 15.80 | 0.054 | | 20.185 ± 0.351 | 2981.00 ± 26.95 | 1.091 |
| DUOL | **11.757 ± 0.237** | 407.50 ± 12.80 | 0.080 | | **19.438 ± 0.282** | 2494.95 ± 26.19 | 1.069 |

| Algorithm | MITFace | | | | w7a | | |
|---|---|---|---|---|---|---|---|
| | Mistake (%) | Support Vectors (#) | Time (s) | | Mistakes (%) | Support Vectors (#) | Time (s) |
| Perceptron | 4.665 ± 0.192 | 325.50 ± 13.37 | **0.207** | | 4.027 ± 0.095 | **994.40 ± 23.57** | 3.392 |
| ROMMA | 4.114 ± 0.155 | **287.05 ± 10.84** | 0.285 | | 4.158 ± 0.087 | 1026.75 ± 21.51 | 1.875 |
| agg-ROMMA | 3.137 ± 0.093 | 1121.15 ± 24.18 | 0.555 | | 3.500 ± 0.061 | 2318.65 ± 60.49 | 3.257 |
| ALMA$_2$(0.9) | 4.467 ± 0.169 | 400.10 ± 10.53 | 0.297 | | 3.518 ± 0.071 | 1031.05 ± 15.33 | **1.314** |
| PA-I | 3.190 ± 0.128 | 1155.45 ± 14.53 | 0.439 | | 3.701 ± 0.057 | 2839.60 ± 41.57 | 2.691 |
| PA-II | 3.108 ± 0.112 | 1222.05 ± 13.73 | 0.463 | | 3.571 ± 0.053 | 3391.50 ± 51.94 | 3.311 |
| DUOL$_{iter}$ | 2.551 ± 0.128 | 963.45 ± 23.80 | 0.572 | | 4.456 ± 0.073 | 3048.85 ± 54.53 | 4.566 |
| DUOL$_{appr}$ | 2.687 ± 0.140 | 1262.50 ± 20.68 | 0.656 | | 3.116 ± 0.104 | 2908.95 ± 28.65 | 3.679 |
| DUOL | **2.151 ± 0.106** | 697.95 ± 13.17 | 0.445 | | **2.914 ± 0.045** | 2402.55 ± 39.88 | 6.470 |

Table 2: Evaluation of online learning algorithms on the binary-class data sets.

we observe that DUOL achieves significantly smaller *mistake rates* than the other single-updating algorithms in all cases. This shows that the proposed double updating approach is effective in improving the performance of online prediction. By examining the number of support vectors, we observed that DUOL results in sparser classifiers than the three aggressive online learning algorithms, and denser classifiers than the three non-aggressive algorithms.

Third, according to the results of running time, we observe that DUOL is overall efficient as compared with the state-of-the-art online learning algorithms. Among all the algorithms in comparison, Perceptron, due to its simplicity nature, is clearly the most efficient algorithm. Since DUOL requires double updates, it is less efficient than PA, ROMMA and ALMA algorithms, but is comparable to the agg-ROMMA algorithm. Note that the comparisons of running time costs are slightly different compared with the results in our previous conference paper (Zhao et al., 2009) because we did some improvements of efficiency for the implementations of some existing algorithms in this journal article.

### 5.3 Evaluation of Different Auxiliary Example Selection Strategies and the Sensitivity to Parameter $C$ for DUOL

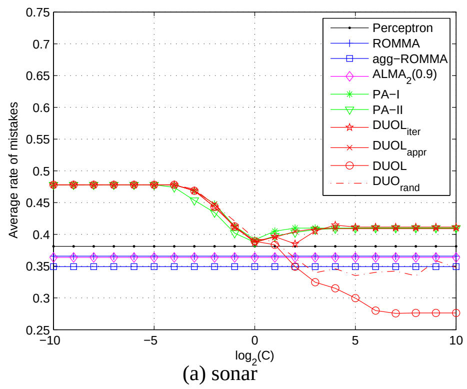
As the performance of DUOL quite relies on the choice of auxiliary examples, in this section, we evaluate different auxiliary example selection strategies. Specifically, we compare the proposed strategy to a random selection approach, referred to as "DUOL$_{rand}$", which randomly chooses an auxiliary example from the existing support vectors. The exact solution to the problem in (1), given by Proposition 2, is used for updating the weights of both examples. We set $\rho = 0$ and $\sigma = 8$ for all the data sets, same as the previous experiments.

Figure 3 compares the online prediction performance between DUOL and DUOL$_{rand}$ as well as the other competing algorithms with varied $C$ values across eight different data sets. Several observations can be drawn from the results.

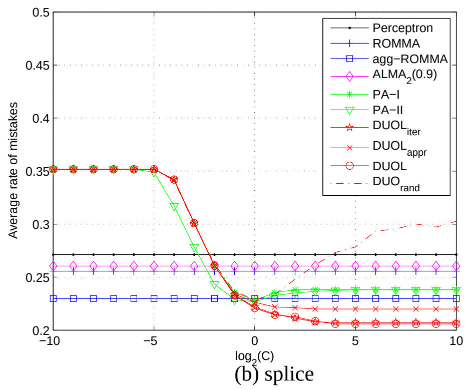
First, it is clear to see that the proposed strategy for selecting auxiliary examples is more effective than the random selection strategy for most cases. Second, among all the compared algorithms, we observe that DUOL always achieves the best performance when $C$ is sufficiently large (e.g., $C > 10$), except for data sets "german" and "w7a" where a smaller $C$ value tends to produce a better result. This observation is consistent to our previous theoretical result, which indicates setting a large $C$ value usually implies more strong updates and consequently a better mistake bound. Third, we observe that the proposed DUOL algorithm is significantly more accurate than the other two variants of double updating online learning algorithms (DUOL$_{iter}$ and DUOL$_{appr}$) for varied $C$ values, as we expected. We observe that DUOL$_{iter}$, the iterative updating approach, performs unstably, which might be due to local optimum suffered from its heuristic update. This observation validates the importance of performing the optimal double updates by the proposed DUOL algorithm.

### 5.4 Empirical Evaluation of Mistake Bounds

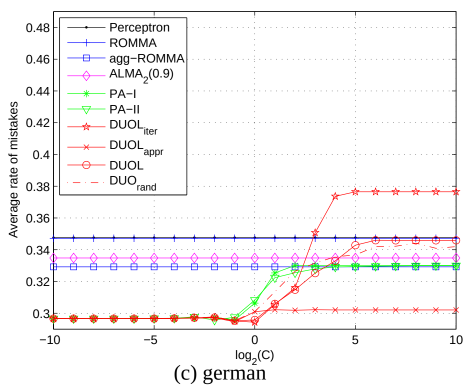
To examine how the double updating strategy affects the mistake bound, we empirically compare $M$, the total number of mistakes made by the DUOL algorithm, $M_d^w(\rho)$, the number of mistake cases where the weak double updates are applied, and $M_d^s(\rho)$, the number of mistake cases where the strong double updates are applied. Figure 4 shows the comparison between $M$, $M_d^w(\rho)$, and $M_d^s(\rho)$ by varying $\rho$ from 0 to 1.



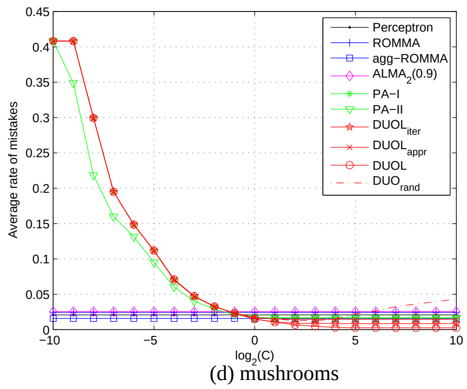
(a) sonar



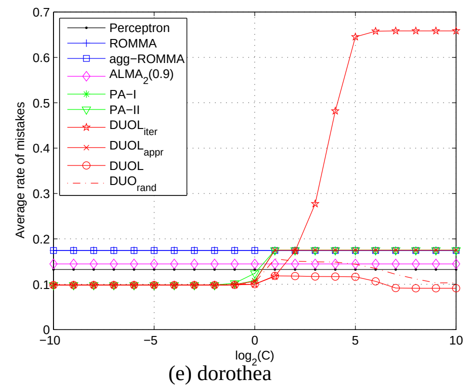
(b) splice



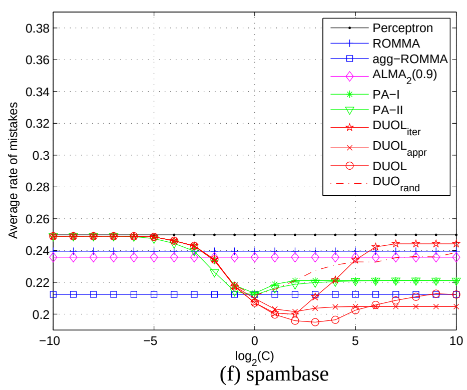
(c) german



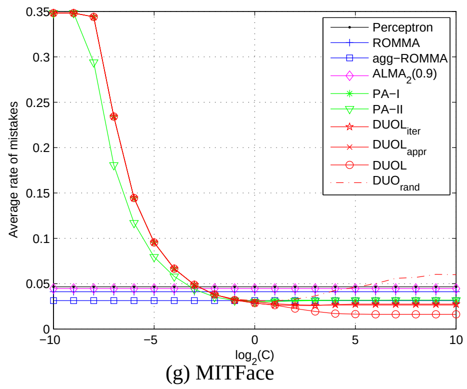
(d) mushrooms



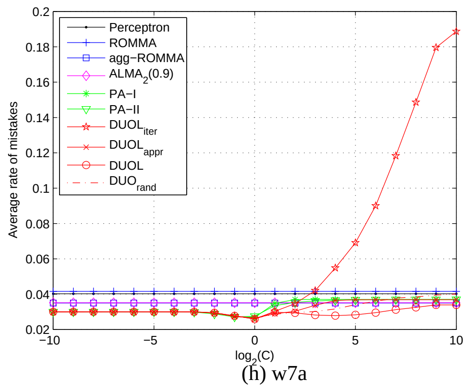
(e) dorothea



(f) spambase



(g) MITFace



(h) w7a

Figure 3: Comparison between DUOL and DUOL$_{rand}$ with varied $C$ values.

First, we observe that double updates are frequently applied when $\rho$ is small. This is because it is easier to find an auxiliary example for double updating when $\rho$ is small. Further, we find that setting $\rho$ close to 0 by default often leads to the best or close to the best results. Second, we observe that the number of weak updates is significantly larger than that of strong updates. This is because the condition of conducting a strong double update is significantly more difficult to be satisfied that that for a weak double update. Third, we observe that both $M_d^w(\rho)$ and $M_d^s(\rho)$ monotonically decrease when increasing the value of $\rho$. In the extreme case, when $\rho$ is close to 1, their value often drops to zero, indicating that no double update was applied. In the meantime, we find that the total number of mistakes often reaches the maximum, as $\rho$ approaches 1. These results again validate the importance and effectiveness of the proposed double updating algorithm.

## 5.5 Testbeds and Experimental Setup for Multiclass Online Learning

Table 3 shows the multiclass data sets from Web machine learning repository used in our experiments. We compare the proposed M-DUOL algorithm with six state-of-the-art online learning algorithms. The first three algorithms are variants of Perceptron-based on methods studied in Crammer and Singer (2003). They are: (i) “Max”, the perceptron method based on the *max-score* multiclass update, (ii) “Uniform”, the perceptron method based on the *Uniform* multiclass update, and (iii) “Prop”, the perceptron method based on the *proportion* multiclass update. We also compare the proposed algorithm with the other three state-of-the-art online multi-class learning algorithms, including the MIRA algorithm proposed by Crammer and Singer (2003), and the Passive-Aggressive (PA) algorithms, “PA-I” and “PA-II” proposed by Crammer et al. (2006). Similar to the experiments of binary classification, we implement three variants of the proposed M-DUOL algorithm based on different solvers to the problem in (5), that is, “M-DUOL$_{appr}$”, “M-DUOL”, and “M-DUOL$_{iter}$”. For all experiments, we use the Gaussian kernel with $\sigma = 8$ and set $C = 10$. The threshold $\rho$ in the proposed algorithms is set to 0 for all experiments. All the experiments were repeated 20 time and the final results are averaged over 20 runs.

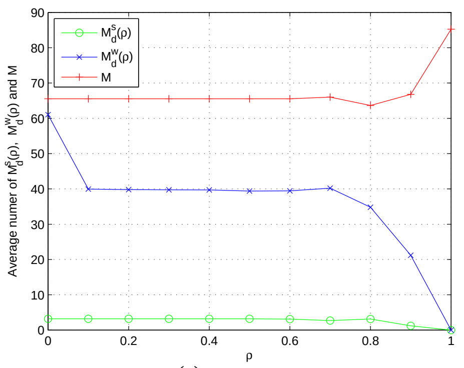
| data set | # training examples | # classes | # features |
|---|---:|---:|---:|
| vehicle | 846 | 4 | 18 |
| dna | 2,000 | 3 | 180 |
| segment | 2,310 | 7 | 19 |
| satimage | 4,435 | 6 | 36 |
| usps | 7,291 | 10 | 256 |
| mnist | 10,000 | 10 | 780 |
| letter | 15,000 | 26 | 16 |
| protein | 17,766 | 3 | 357 |

Table 3: Multiclass data sets used in the experiments.

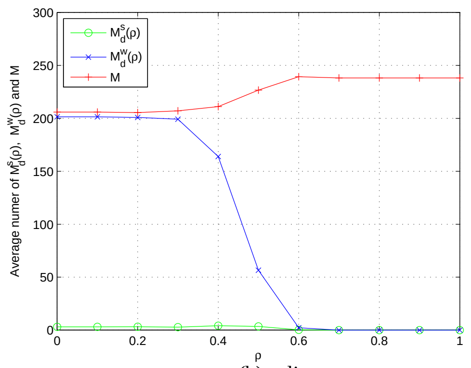
## 5.6 Performance Evaluation for Multi-class Online Learning

Table 4 summarizes the empirical performance for multi-class online learning. Several observations can be drawn from the experimental results.

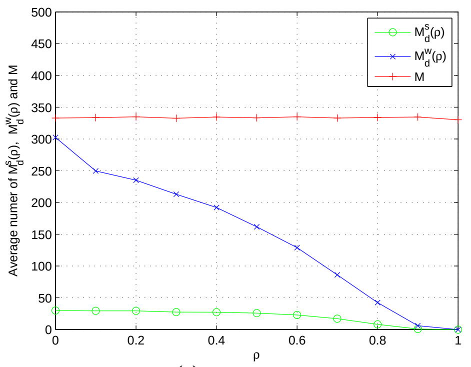
First, by comparing all the baseline algorithms, we find that the two PA algorithms yield considerably lower mistake rates than the other single-updating online learning algorithms. On the other hand, the classifiers learned by the three Perceptron-based algorithms (Max, Uniform, and



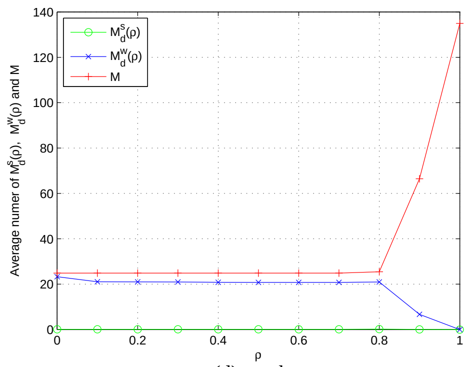
(a) sonar



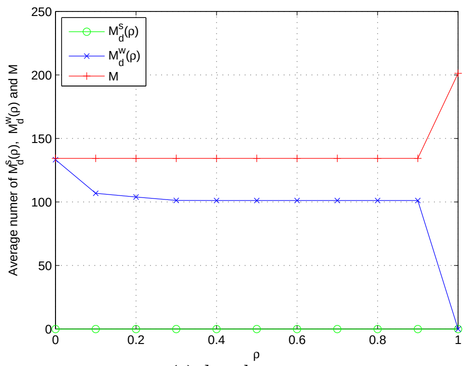
(b) splice



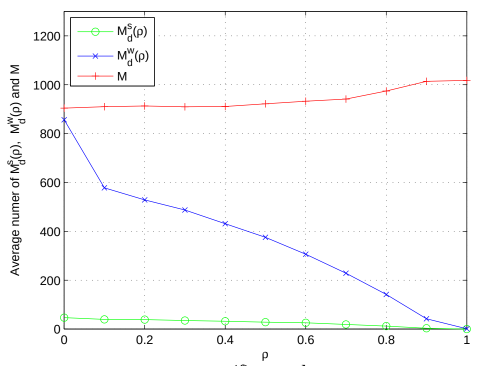
(c) german



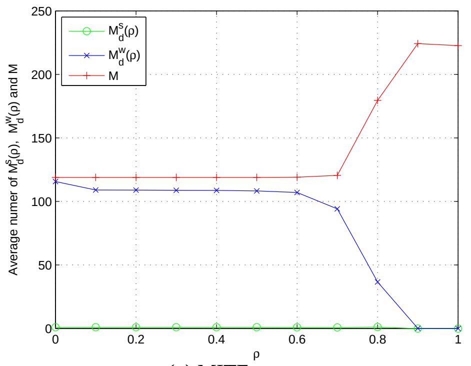
(d) mushrooms



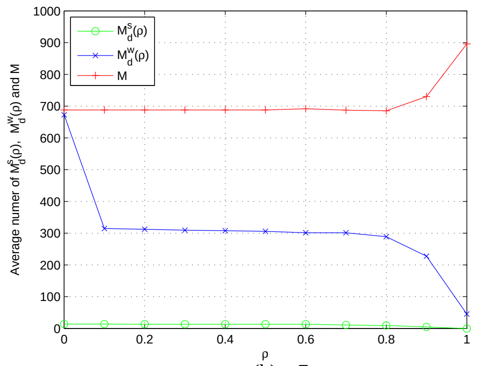
(e) dorothea



(f) spambase



(g) MITFace



(h) w7a

Figure 4: Empirical comparison of $M$, $M_d^w(\rho)$ and $M_d^s(\rho)$ w.r.t. varied $\rho \in [0,1]$ values.

| Algorithm | vehicle | | | dna | | |
|---|---|---|---|---|---|---|
| | Mistake (%) | Support Vectors (#) | Time (s) | Mistakes (%) | Support Vectors (#) | Time (s) |
| Max | 64.882 ± 1.643 | 548.90 ± 13.90 | **0.079** | 20.460 ± 0.770 | 409.20 ± 15.41 | **0.192** |
| Uniform | 65.934 ± 1.554 | 557.80 ± 13.15 | 0.109 | 19.875 ± 0.427 | **397.50 ± 8.54** | 0.264 |
| Prop | 66.678 ± 1.757 | 564.10 ± 14.86 | 0.116 | 20.268 ± 0.555 | 405.35 ± 11.10 | 0.267 |
| MIRA | 62.252 ± 2.114 | **526.65 ± 17.89** | 1.821 | 26.920 ± 0.880 | 538.40 ± 17.61 | 5.304 |
| PA-I | 67.086 ± 1.479 | 781.70 ± 12.42 | 0.091 | 15.503 ± 0.474 | 1224.35 ± 13.48 | 0.326 |
| PA-II | 66.909 ± 1.475 | 789.30 ± 10.73 | 0.089 | 15.398 ± 0.467 | 1237.50 ± 13.12 | 0.325 |
| M-DUOL$_{iter}$ | 70.674 ± 1.194 | 758.05 ± 8.65 | 0.162 | 11.668 ± 0.599 | 1086.00 ± 16.39 | 0.502 |
| M-DUOL$_{appr}$ | 69.634 ± 1.463 | 828.05 ± 4.48 | 0.158 | 14.105 ± 0.611 | 1281.75 ± 14.44 | 0.495 |
| M-DUOL | **51.950 ± 1.948** | 719.25 ± 10.95 | 0.172 | **10.340 ± 0.513** | 869.80 ± 12.61 | 0.438 |
| Algorithm | segment | | | satimage | | |
| | Mistake (%) | Support Vectors (#) | Time (s) | Mistakes (%) | Support Vectors (#) | Time (s) |
| Max | 41.342 ± 1.013 | 955.00 ± 23.40 | **0.414** | 29.628 ± 0.561 | 1314.00 ± 24.89 | **0.826** |
| Uniform | 41.468 ± 0.550 | 957.90 ± 12.71 | 0.566 | 28.440 ± 0.398 | 1261.30 ± 17.64 | 1.071 |
| Prop | 41.589 ± 0.714 | 960.70 ± 16.48 | 0.565 | 28.878 ± 0.467 | 1280.75 ± 20.72 | 1.087 |
| MIRA | 35.784 ± 3.770 | **826.55 ± 87.08** | 9.193 | 27.536 ± 2.228 | **1221.20 ± 98.80** | 15.229 |
| PA-I | 39.775 ± 0.665 | 1852.75 ± 19.90 | 0.573 | 27.377 ± 0.361 | 2676.40 ± 24.88 | 1.296 |
| PA-II | 39.842 ± 0.655 | 1870.70 ± 18.97 | 0.577 | 27.258 ± 0.429 | 2709.50 ± 23.77 | 1.307 |
| M-DUOL$_{iter}$ | 41.416 ± 1.084 | 1787.90 ± 31.00 | 0.903 | 33.894 ± 0.567 | 2787.45 ± 43.18 | 2.024 |
| M-DUOL$_{appr}$ | 39.314 ± 0.791 | 1923.60 ± 14.31 | 0.871 | 26.222 ± 0.464 | 3052.50 ± 31.39 | 2.024 |
| M-DUOL | **20.580 ± 0.705** | 1265.15 ± 28.39 | 0.693 | **22.524 ± 0.482** | 2066.85 ± 32.99 | 1.505 |
| Algorithm | usps | | | mnist | | |
| | Mistake (%) | Support Vectors (#) | Time (s) | Mistakes (%) | Support Vectors (#) | Time (s) |
| Max | 10.025 ± 0.195 | 730.90 ± 14.21 | **1.459** | 15.318 ± 0.168 | 1531.80 ± 16.80 | **2.744** |
| Uniform | 9.445 ± 0.150 | **688.60 ± 10.91** | 1.858 | 14.603 ± 0.201 | **1460.25 ± 20.15** | 3.631 |
| Prop | 9.614 ± 0.176 | 700.95 ± 12.86 | 1.868 | 14.763 ± 0.228 | 1476.30 ± 22.78 | 3.635 |
| MIRA | 11.572 ± 0.403 | 843.75 ± 29.39 | 44.663 | 18.037 ± 0.539 | 1803.70 ± 53.93 | 67.168 |
| PA-I | 6.641 ± 0.158 | 2528.45 ± 23.48 | 2.669 | 11.026 ± 0.208 | 4773.70 ± 32.84 | 5.771 |
| PA-II | 6.568 ± 0.116 | 2561.95 ± 27.94 | 2.606 | 10.959 ± 0.238 | 4830.40 ± 27.06 | 5.824 |
| M-DUOL$_{iter}$ | 5.743 ± 0.158 | 2284.15 ± 40.06 | 3.160 | 8.947 ± 0.182 | 4398.95 ± 46.46 | 9.031 |
| M-DUOL$_{appr}$ | 6.002 ± 0.132 | 2725.40 ± 23.55 | 3.541 | 9.640 ± 0.164 | 5163.05 ± 37.34 | 10.386 |
| M-DUOL | **5.162 ± 0.149** | 1759.30 ± 23.44 | 2.408 | **8.282 ± 0.183** | 3557.15 ± 25.17 | 7.050 |
| Algorithm | letter | | | protein | | |
| | Mistake (%) | Support Vectors (#) | Time (s) | Mistakes (%) | Support Vectors (#) | Time (s) |
| Max | 71.562 ± 0.538 | 10734.35 ± 80.63 | **18.749** | 47.657 ± 0.221 | 8466.75 ± 39.21 | **12.842** |
| Uniform | 71.973 ± 0.280 | 10795.90 ± 41.99 | 47.031 | 46.828 ± 0.272 | **8319.45 ± 48.36** | 14.342 |
| Prop | 72.033 ± 0.273 | 10804.95 ± 40.89 | 43.683 | 47.260 ± 0.260 | 8396.15 ± 46.13 | 14.620 |
| MIRA | 67.709 ± 1.196 | **10156.35 ±179.54** | 467.019 | 47.905 ± 0.922 | 8510.80 ±163.74 | 42.174 |
| PA-I | 72.283 ± 0.338 | 14708.55 ± 15.27 | 24.848 | 47.657 ± 0.230 | 14153.25 ± 49.06 | 23.409 |
| PA-II | 72.339 ± 0.380 | 14735.55 ± 15.86 | 24.131 | 47.550 ± 0.285 | 14285.85 ± 44.94 | 23.602 |
| M-DUOL$_{iter}$ | 73.066 ± 0.326 | 14614.65 ± 22.26 | 210.684 | 50.070 ± 0.392 | 14191.85 ± 64.80 | 55.622 |
| M-DUOL$_{appr}$ | 69.992 ± 0.331 | 14892.70 ± 11.77 | 215.587 | 51.459 ± 0.582 | 16000.55 ± 72.07 | 63.065 |
| M-DUOL | **54.068 ± 0.351** | 13140.40 ± 37.33 | 186.452 | **46.281 ± 0.418** | 12550.10 ± 87.27 | 43.774 |

Table 4: Evaluation of multiclass online learning algorithms on the multiclass data sets.

Prop) and MIRA are considerably sparser than those learned by the two PA algorithms. We believe that this can be attributed to the aggressive updating strategies used by the PA algorithms. Second, among the three variants of double updating for multi-label learning, it is not surprising to observe that M-DUOL yields the lowest mistake rates for all data sets. Further, among all the algorithms, we observe that the M-DUOL algorithm makes the least number of mistakes for all data sets, and significantly outperforms all the baseline algorithms.

Second, by examining the sparsity of classifiers learned by the proposed algorithms, we observe that the number of support vectors identified by M-DUOL is usually smaller than that of the PA algorithms (except for data set "vehicle"), but is significantly larger than those of the four non-aggressive algorithms (i.e., Max, Uniform, Prop, and MIRA).

Finally, comparing the running time cost, we observe that the Max algorithm is the most efficient one, while MIRA is the least efficient approach for all the data sets. Despite the additional time needed for double updates, overall we found that the running time of the proposed M-DUOL algorithm is comparable to those of the two PA algorithms (except for the "letter" data set where the time costs of the M-DUOL algorithms are considerably greater than those of the PA algorithms).

## 6. Discussions and Future Directions

Although encouraging results have been achieved by the proposed novel DUOL algorithms, we should address the limitations of our current work and discuss some research directions for future improvements. First of all, the proposed DUOL algorithm is based on the Passive Aggressive online learning algorithms (Crammer et al., 2006). For the future work, it is possible to extend other single update online learning methods, such as EG (Kivinen and Warmuth, 1995), for double updating. Second, the approach for choosing an auxiliary example from existing support vectors may be further improved by exploring the heuristics for measuring the informativeness of an example. Finally, we plan to extend the proposed double updating framework for budget online learning to make sparse classifiers.

## 7. Conclusions

This paper presented a novel "double updating" approach to online learning named as "DUOL", which not only updates the weight of the misclassified example, but also adjusts the weight of one existing support vector that the most seriously conflicts with the new support vector. We show that the mistake bound for an online classification task can be significantly reduced by the proposed DUOL algorithms. We have conducted an extensive set of experiments by comparing with a number of algorithms for both binary and multiclass online classifications. Promising empirical results showed that the proposed double updating online learning algorithms consistently outperform the single-update online learning algorithms.

## Acknowledgments

The work described in this paper was supported in part by Singapore MOE ARC Tier-2 Grant (T208B2203), Singapore MOE ARC Tier-1 Grant (RG67/07), National Science Foundation (IIS-0643494), and US Office of Naval Research (ONR N00014-09-1-0663).

## Appendix A. The Proof for Proposition 2

**Proof** The optimization (1) can be rewritten to the following equivalent optimization:

$$\min_{\gamma_a, d_{\gamma_b}} \quad \frac{k_a}{2}\gamma_a^2 + \frac{k_b}{2}d_{\gamma_b}^2 + w_{ab}\gamma_a d_{\gamma_b} - \ell_a\gamma_a - \ell_b d_{\gamma_b},$$

$$\text{s.t.} \quad \gamma_a - C \le 0, \tag{6}$$

$$-\gamma_a \le 0, \tag{7}$$

$$d_{\gamma_b} - C + \hat{\gamma}_b \le 0, \tag{8}$$

$$-d_{\gamma_b} - \hat{\gamma}_b \le 0, \tag{9}$$

where $k_a, k_b > 0$, $w_{ab} \le 0$, $\ell_a = 1 - y_a f(x_a) \ge 0$, $\ell_b = 1 - y_b f(x_b) \ge 0$ and $\hat{\gamma}_b > 0$. With $\lambda_1$, $\lambda_2$, $\lambda_3$ and $\lambda_4$ as Lagrange multipliers, the KKT conditions for this problem consist of the constraints (6)-(9), the nonnegativity constraints $\lambda_i \ge 0$, $\forall i$, the complementary slackness conditions

$$\lambda_1(\gamma_a - C) = 0,\ \lambda_2(-\gamma_a) = 0,\ \lambda_3(d_{\gamma_b} - C + \hat{\gamma}_b) = 0,\ \lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$$

and zero gradient conditions:

$$k_a\gamma_a + w_{ab}d_{\gamma_b} - \ell_a + \lambda_1 - \lambda_2 = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}\gamma_a - \ell_b + \lambda_3 - \lambda_4 = 0.$$

We will discuss every possible condition to compute the closed-form solution. Firstly, we will discuss the case $\lambda_1 \ne 0$:

### A.1 Case 1. If $\lambda_1 \ne 0$

Since $\lambda_1(\gamma_a - C) = 0$, we have $\gamma_a = C$; further, because $\lambda_2(-\gamma_a) = 0$, we have $\lambda_2 = 0$. Under the condition $\lambda_1 \ne 0$, we will discuss $\lambda_3 \ne 0$ and $\lambda_3 = 0$ separately as follows:

#### A.1.1 SUB-CASE 1.1. IF $\lambda_3 \ne 0$

Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, we have $d_{\gamma_b} = C - \hat{\gamma}_b$, as a result $\lambda_4(-C) = 0$, so $\lambda_4 = 0$. Plugging the results $\gamma_a = C$, $\lambda_2 = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and $\lambda_4 = 0$ into the zero gradient condition, we have

$$k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b + \lambda_3 = 0.$$

Thus, we have

$$\lambda_1 = -[k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a] \quad \text{and} \quad \lambda_3 = -[k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b].$$

As a result, if

$$-(k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) > 0 \quad \text{and} \quad -(k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) > 0,$$

then KKT conditions are satisfied, $(\gamma_a, d_{\gamma_b}) = (C, C - \hat{\gamma}_b)$ is the unique solution.

#### A.1.2 SUB-CASE 1.2. IF $\lambda_3 = 0$

When $\lambda_3 = 0$, we only conclude $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$.
Under the conditions $\lambda_1 \neq 0$ and $\lambda_3 = 0$, we will discuss the two cases $\lambda_4 \neq 0$ and $\lambda_4 = 0$, respectively as follows.
*Sub-case 1.2.1. If* $\lambda_4 \neq 0$. Since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, we have $d_{\gamma_b} = -\hat{\gamma}_b$. Plugging the results $\lambda_2 = 0$, $\gamma_a = C$, $\lambda_3 = 0$ and $d_{\gamma_b} = -\hat{\gamma}_b$ in to the zero gradient conditions:

$$k_a C + w_{ab}(-\hat{\gamma}_b) - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) + w_{ab}C - \ell_b - \lambda_4 = 0.$$

But since $k_b(-\hat{\gamma}_b) < 0$ $w_{ab}C \leq 0$ and $\ell_b, \lambda_4 \geq 0$, $k_b(-\hat{\gamma}_b) + w_{ab}C - \ell_b - \lambda_4 < 0$, which contradicts the equation above.
*Sub-case 1.2.2. If* $\lambda_4 = 0$. Plugging the conditions $\gamma_a = C$, $\lambda_2 = 0$, $\lambda_3 = 0$ and $\lambda_4 = 0$ into the zero gradient equations:

$$k_a C + w_{ab} d_{\gamma_b} - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}C - \ell_b = 0.$$

Solving the above equations leads to the following:

$$\lambda_1 = \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} \quad \text{and} \quad d_{\gamma_b} = \frac{\ell_b - w_{ab}C}{k_b}.$$

If $\frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0$ and $\frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$, then the KKT conditions are all satisfied; as a result, $(\gamma_a, d_{\gamma_b}) = (C, \frac{\ell_b - w_{ab}C}{k_b})$ is the unique optimal solution.

Next we will discuss the situation with the condition $\lambda_1 = 0$.

### A.2 Case 2. If $\lambda_1 = 0$

Under the condition $\lambda_1 = 0$, we only conclude $\gamma_a \in [0, C]$. We will discuss the cases $\lambda_2 \neq 0$ and $\lambda_2 = 0$ under the condition $\lambda_1 = 0$, respectively.

#### A.2.1 SUB-CASE 2.1. IF $\lambda_2 \neq 0$

Since $\lambda_2(-\gamma_a) = 0$, we conclude $\gamma_a = 0$. Under the conditions $\lambda_1 = 0$ and $\lambda_2 \neq 0$, we will discuss the cases $\lambda_3 \neq 0$ and $\lambda_3 = 0$:
*Sub-case 2.1.1. If* $\lambda_3 \neq 0$. Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, plugging the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and $\lambda_4 = 0$ into the zero gradient conditions:

$$w_{ab}(C - \hat{\gamma}_b) - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) - \ell_b + \lambda_3 = 0.$$

Since $w_{ab} \leq 0, C - \hat{\gamma}_b \geq 0$ and $\ell_a \geq 0$, we conclude

$$\lambda_2 = w_{ab}(C - \hat{\gamma}_b) - \ell_a \leq 0.$$

But $\lambda_2 \geq 0$ and $\lambda_2 \neq 0$, conclude $\lambda_2 > 0$, which contradicts the inequality above.
*Sub-case 2.1.2. If* $\lambda_3 = 0$. Under these known conditions, we only know $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$. Below, we will discuss the cases $\lambda_4 \neq 0$ and $\lambda_4 = 0$, under the conditions $\lambda_1 = 0$, $\lambda_2 \neq 0$ and $\lambda_3 = 0$.

- If $\lambda_4 \neq 0$, since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$. From the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $\lambda_3 = 0$ and $d_{\gamma_b} = -\hat{\gamma}_b$ and the zero gradient conditions, we have

  $$w_{ab}(-\hat{\gamma}_b) - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) - \ell_b - \lambda_4 = 0.$$

  Since $k_b, \hat{\gamma}_b > 0$ and $\ell_b \geq 0$, we conclude

  $$\lambda_4 = k_b(-\hat{\gamma}_b) - \ell_b < 0.$$

  But the equation above contradicts $\lambda_4 > 0$.

- Else if $\lambda_4 = 0$, from the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $\lambda_3 = 0$ and $\lambda_4 = 0$ and the zero gradient conditions, we have

  $$w_{ab}d_{\gamma_b} - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b d_{\gamma_b} - \ell_b = 0.$$

  Since $w_{ab} \leq 0$, $\ell_b, \ell_a \geq 0$ and $k_b > 0$,

  $$\lambda_2 = w_{ab}\frac{\ell_b}{k_b} - \ell_a \leq 0,$$

  which contradicts $\lambda_2 > 0$ (Since $\lambda_2 \neq 0$).

### A.2.2 SUB-CASE 2.2. IF $\lambda_2 = 0$

Under the conditions $\lambda_1 = \lambda_2 = 0$, we only know $\gamma_a \in [0, C]$. Below, we will discuss the two cases $\lambda_3 \neq 0$ and $\lambda_3 = 0$, under the conditions $\lambda_1 = \lambda_2 = 0$.
*Sub-case 2.2.1. If* $\lambda_3 \neq 0$. Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$, as a result $\lambda_4(-C) = 0$, so $\lambda_4 = 0$. From the conditions $\lambda_1 = \lambda_2 = \lambda_4 = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and zero gradient conditions:

$$k_a\gamma_a + w_{ab}(C - \hat{\gamma}_b) - \ell_a = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) + w_{ab}\gamma_a - \ell_b + \lambda_3 = 0.$$

As a result, if

$$\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \quad \text{and} \quad \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0,$$

the unique optimal solution is $(\gamma_a, d_{\gamma_b}) = (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b)$.
*Sub-case 2.2.2. If* $\lambda_3 = 0$. According to $\lambda_1 = \lambda_2 = \lambda_3 = 0$, we only conclude $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$.

- If $\lambda_4 \neq 0$, since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$. From $\lambda_1 = \lambda_2 = \lambda_3 = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$ and zero gradient conditions:

  $$k_a\gamma_a + w_{ab}(-\hat{\gamma}_b) - \ell_a = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) + w_{ab}\gamma_a - \ell_b - \lambda_4 = 0,$$

  since $\lambda_4 = k_b(-\hat{\gamma}_b) + w_{ab}\gamma_a - \ell_b < 0$ which contradicts with the condition $\lambda_4 > 0$.

- If $\lambda_4 = 0$, from $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$ and zero gradient conditions:

  $$k_a\gamma_a + w_{ab}d_{\gamma_b} - \ell_a = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}\gamma_a - \ell_b = 0.$$

  As a result, if $\gamma_a$ and $d_{\gamma_b}$ satisfy the following:

  $$\gamma_a = \frac{k_b\ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2} \in [0, C] \quad \text{and} \quad d_{\gamma_b} = \frac{k_a\ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b],$$

  then $(\gamma_a, d_{\gamma_b}) = (\frac{k_b\ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a\ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2})$ is the unique optimal solution.

*Summary:* The final closed-form solution to the optimization is summarized as:

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C-\hat{\gamma}_b) & \text{if } (k_aC + w_{ab}(C-\hat{\gamma}_b) - \ell_a) < 0 \text{ and } (k_b(C-\hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \text{ and } \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C-\hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{k_a}, C-\hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{k_a} \in [0,C] \text{ and } \ell_b - k_b(C-\hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b\ell_a - w_{ab}\ell_b}{k_ak_b - w_{ab}^2}, \frac{k_a\ell_b - w_{ab}\ell_a}{k_ak_b - w_{ab}^2}) & \text{if } (\frac{k_b\ell_a - w_{ab}\ell_b}{k_ak_b - w_{ab}^2}, \frac{k_a\ell_b - w_{ab}\ell_a}{k_ak_b - w_{ab}^2}) \in [0,C] \times [-\hat{\gamma}_b, C-\hat{\gamma}_b] \end{cases}.$$

■

## Appendix B. The Proof for Proposition 3

**Proof** First of all, the product $H(Y_a) \cdot H(Y_b)$ can be simplified as:

$$H(Y_a) \cdot H(Y_b) = \sum_{i=1}^{k} \sigma(i,a)\sigma(i,b) = \sigma(r_a,a)\sigma(r_a,b) + \sigma(s_a,a)\sigma(s_a,b) = \sigma(r_a,b) - \sigma(s_a,b).$$

We can check the value of $\sigma(r_a,b) - \sigma(s_a,b)$ by examining all possible cases as follows:

1 If $r_a = r_b$ that implies that $x_a$ and $x_b$ have the same relevant labels, then we should have $H(Y_a) \cdot H(Y_b) = 1 - \sigma(s_a,b) \geq 1$ (either 1 or 2);

2 If $r_a \neq r_b$, then:

2.1 If $r_a = s_b$, then $H_{Y_a} \cdot H_{Y_b} = \sigma(s_b,b) - \sigma(s_a,b) = -1 - \sigma(s_a,b) \leq -1$;

2.2 If $r_a \neq s_b$, then $H_{Y_a} \cdot H_{Y_b} = \sigma(r_a,b) - \sigma(s_a,b) = 0 - \sigma(s_a,b)$:

2.2.1 If $s_a = s_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(s_b,b) = 1$;

2.2.2 If $s_a = r_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(r_b,b) = -1$;

2.2.3 If $s_a \neq s_b$ and $s_a \neq r_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(s_a,b) = 0$.

We thus have the fact that $H(Y_a) \cdot H(Y_b) < 0$ holds if and only if ($r_a = s_b$) or ($s_a = r_b$). ■

## Appendix C. The Proof of Proposition 4

In this appendix, we will derive the dual ascent by the multiclass double updating approach. Our approach to the proofs is mainly inspired by the study in Shalev-Shwartz (2007), but our problem is different from their study.

For the convenience of our presentation, we introduce the following notation for our derivation. We denote the loss function for a training example $(x,Y)$ as follows:

$$g(\bar{f}) = \ell\big(\bar{f};(x,Y)\big) = \max_{r\in Y, s\notin Y} \Big[1 - \big(f_r(x) - f_s(x)\big)\Big]_+ .$$

We order all the classes $r$ in the assigned set $Y$ as $r_1, \cdots, r_{\|Y\|}$, and the class $s$ in the unassigned set $\mathcal{Y} \setminus Y$ as $s_1, \cdots, s_{\|[k]/Y\|}$. We slightly abuse our notations by simplifying $\langle f,g\rangle_{\mathcal{H}_K}$ as $\langle f,g\rangle$ and $\|f\|_{\mathcal{H}_K}$ as $\|f\|$ when there is no ambiguity about the space for computing dot product and norm.

We first give a lemma that shows the Fenchel conjugate of the above loss function $g$.

**Lemma 2** *Let $\mathcal{Y} = [k]$ be the possible labels set. $Y \subseteq \mathcal{Y}$ is relevant labels set for $x \in R^n$. $\bar{f} = (f_1, \cdots, f_k)^T$, where $\forall i \in [k]$, $f_i \in \mathcal{H}_\kappa$. And the loss function is defined as follows:*

$$g(\bar{f}) = \max_{r \in Y, s \notin Y} \Big[1 - \big(f_r(x) - f_s(x)\big)\Big]_+.$$

*Then for any $\bar{\lambda} = (\lambda_1, \cdots, \lambda_k)^T$, where $\forall i\ \lambda_i \in \mathcal{H}_\kappa$, we have g's Fenchel conjugate as:*

$$g^*(\bar{\lambda}) = \begin{cases} -\sum_{i,j} \alpha_{ij} & \text{if } \lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) = 0 \text{ and } \lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) = 0 \\ \infty & \text{otherwise} \end{cases},$$

*where $\bar{\alpha} = (\alpha_{ij}) \in \mathcal{A} = [A | A \in R_+^{\|Y\|} \times R_+^{(k-\|Y\|)}, \|A\|_1 \le 1]$ and $(r_i \times s_j) \in \mathcal{B} = Y \times ([k]/Y)$.*

**Proof** The approach of our proof is similar to the method for proving the “Max-of-hinge” in Shalev-Shwartz (2007). First of all, it is not difficult to show that the loss function can be re-formulated as follows:

$$\begin{aligned} g(\bar{f}) &= \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(f_{r_i}(x) - f_{s_j}(x)\big)\Big] \\ &= \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle f_{r_i}(\cdot), \kappa(x,\cdot)\rangle - \langle f_{s_j}(\cdot), \kappa(x,\cdot)\rangle\big)\Big]. \end{aligned}$$

As a result, we have:

$$\begin{aligned} g^*(\bar{\lambda}) &= \max_{\bar{f}} \Big\{ \langle \bar{\lambda}, \bar{f} \rangle - g(\bar{f}) \Big\} \\ &= \max_{\bar{f}} \Big\{ \sum_{n=1}^{k} \langle \lambda_n, f_n \rangle - \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle f_{r_i}(\cdot), \kappa(x,\cdot)\rangle - \langle f_{s_j}(\cdot), \kappa(x,\cdot)\rangle\big)\Big] \Big\}. \end{aligned}$$

For any $f_n, \lambda_n \in \mathcal{H}_\kappa$, they can be written as: $f_n = \beta_n \kappa(x,\cdot) + f_n^\perp, \lambda_n = \gamma_n \kappa(x,\cdot) + \lambda_n^\perp$, where $f_n^\perp, \lambda_n^\perp \in \mathcal{V}^\perp$, $\mathcal{V} = \overline{span\{\kappa(x,\cdot)\}}$. As a result, we have

$$g^*(\bar{\lambda}) = \max_{\bar{f}} \Big\{ \sum_{n=1}^{k} \Big(\langle \lambda_n^\perp, f_n^\perp \rangle + \beta_n \gamma_n \kappa(x,x)\Big) - \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\beta_{r_i}\kappa(x,x) - \beta_{s_j}\kappa(x,x)\big)\Big] \Big\}.$$

When $\lambda_n^\perp \neq 0$, the $\max_{f^\perp} \langle \lambda_n^\perp, f_n^\perp \rangle$ will be $\infty$, resulting $g^*(\bar{\lambda}) = \infty$. Otherwise, if $\lambda_n^\perp = 0$, $\forall n$, the term $f_n^\perp$ does not take effect for the objective; as a result, the optimal $f_n$ can be written in the form of $\beta_n \kappa(x,\cdot)$ and the conjugate is computed as follows:

$$\begin{aligned} g^*(\bar{\lambda}) &= \max_{\bar{\beta}_n} \Big\{ \sum_{n=1}^{k} \beta_n \gamma_n \kappa(x,x) - \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\beta_{r_i}\kappa(x,x) - \beta_{s_j}\kappa(x,x)\big)\Big] \Big\} \\ &= \max_{\bar{\beta}_n} \min_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \Big\{ \sum_{n=1}^{k} \langle \lambda_n, \beta_n \kappa(x,\cdot) \rangle - \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle \beta_{r_i}\kappa(x,\cdot), \kappa(x,\cdot)\rangle - \langle \beta_{s_j}\kappa(x,\cdot), \kappa(x,\cdot)\rangle\big)\Big] \Big\} \\ &= \min_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \max_{\bar{\beta}_n} \Big\{ \sum_{n=1}^{k} \langle \lambda_n, \beta_n \kappa(x,\cdot) \rangle - \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle \beta_{r_i}\kappa(x,\cdot), \kappa(x,\cdot)\rangle - \langle \beta_{s_j}\kappa(x,\cdot), \kappa(x,\cdot)\rangle\big)\Big] \Big\} \\ &= \min_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \Big\{ -\sum_{i,j} \alpha_{ij} + \max_{\bar{\beta}_n} \Big[ \sum_{r_i} \langle \beta_{r_i}\kappa(x,\cdot), \lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) \rangle + \sum_{s_j} \langle \beta_{s_j}\kappa(x,\cdot), \lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) \rangle \Big] \Big\}. \end{aligned}$$

The fourth equality is guaranteed by the strong max-min property (Boyd and Vandenberghe, 2004), and more importantly, we can see that only when $\bar{\alpha}$ satisfies $\lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) = 0$ and $\lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) = 0$, the second term in the equation above will be zero; otherwise, it will be $\infty$. Therefore, we have the resulting Fenchel conjugate of $g(\bar{f})$ as follows:

$$g^*(\bar{\lambda}) = \begin{cases} -\sum_{i,j}\alpha_{ij} & \lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) = 0 \text{ and } \lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) = 0 \\ \infty & \text{otherwise} \end{cases} .$$

■

Given the above Fenchel dual of loss function, we can derive the dual for the optimization problem given on the right-hand side of Equation (4), as given in the following lemma.

**Lemma 3** *Suppose the complexity measure function is given as $F(\bar{f}) = \sum_{i=1}^k \frac{1}{2}\|f_i\|^2_{\mathcal{H}_\kappa}$, and we set $\alpha_{ij}$ to zeros for $\forall(i,j) \in \{[Y_t \times ([k]/Y_t)]/(r_t,s_t)\}$, where $(r_t,s_t)$ is defined in Equation (2). Then the dual objective function for optimization given on the right-hand side of Equation (4) can be expressed as follows:*

$$D(\gamma_1,\cdots,\gamma_T) = -\sum_{i=1}^k \frac{1}{2}\|\sum_{t=1}^T \sigma(i,t)\gamma_t\kappa(x_t,\cdot)\|^2 + \sum_{t=1}^T \gamma_t,$$

*where* $\gamma_t \in [0,C]$ *and* $\sigma(i,t) = \begin{cases} 1 & \textit{if } i = r_t \\ -1 & \textit{if } i = s_t \\ 0 & \textit{otherwise} \end{cases} .$

**Proof** The proof here resembles the one in the section 3.2 of Shalev-Shwartz (2007). Firstly, we note that the problem (4) is equivalent to the following:

$$\inf_{\bar{f}_0,\bar{f}_1,\cdots,\bar{f}_T} \left(F(\bar{f}_0) + \sum_{t=1}^T Cg_t(\bar{f}_t)\right) \text{ s.t. } \bar{f}_0, \bar{f}_t \in \bar{\mathcal{H}}_\kappa \text{ and } \forall t \in [T], \bar{f}_t = \bar{f}_0.$$

By introducing $T$ function vectors $\bar{\lambda}_1,\cdots,\bar{\lambda}_T$, in which each $\bar{\lambda}_t = (\lambda_{t,1},\cdots,\lambda_{t,k}) \in \bar{H}_\kappa$ is a Lagrange multipliers for the constraint $\bar{f}_t = \bar{f}_0$, we can obtain the following Lagrangian:

$$\mathcal{L}(\bar{f}_0,\cdots,\bar{f}_T,\bar{\lambda}_1,\cdots,\bar{\lambda}_T) = F(\bar{f}_0) + \sum_{t=1}^T Cg_t(\bar{f}_t) + \sum_{t=1}^T \langle\bar{\lambda}_t, \bar{f}_0 - \bar{f}_t\rangle.$$

The dual objective function can be derived as follows:

$$\begin{aligned} D(\bar{\lambda}_1,\cdots,\bar{\lambda}_T) &= \inf_{\bar{f}_0,\bar{f}_1,\cdots,\bar{f}_T} \mathcal{L}(\bar{f}_0,\cdots,\bar{f}_T,\bar{\lambda}_1,\cdots,\bar{\lambda}_T) \\ &= -\sup_{\bar{f}_0}\left[\langle\bar{f}_0, -\sum_{t=1}^T \bar{\lambda}_t\rangle - F(\bar{f}_0)\right] - \sum_{t=1}^T \sup_{\bar{f}_t}\left[\langle\bar{f}_t,\bar{\lambda}_t\rangle - Cg_t(\bar{f}_t)\right] \\ &= -F^*(-\sum_{t=1}^T \bar{\lambda}_t) - \sum_{t=1}^T (Cg_t)^*(\bar{\lambda}_t) = -F^*(-\sum_{t=1}^T \bar{\lambda}_t) - \sum_{t=1}^T Cg_t^*(\frac{\bar{\lambda}_t}{C}). \end{aligned}$$

Because $F(\bar{f}) = \sum_{i=1}^k \frac{1}{2}\|f_i\|^2_{\mathcal{H}_\kappa}$, we have $F^* = F$. The dual problem thus becomes:

$$D(\bar{\lambda}_1,\cdots,\bar{\lambda}_T) = -\sum_{i=1}^k \frac{1}{2}\|-\sum_{t=1}^T \lambda_{t,i}\|^2_{\mathcal{H}_\kappa} - \sum_{t=1}^T Cg_t^*(\frac{\bar{\lambda}_t}{C}).$$

Because we want to maximize the dual objective, according to Lemma 2, we should set

$$\frac{\lambda_{t,r_i^t}}{C}+\sum_j \alpha_{ij}^t k(x_t,\cdot)=0, \quad \frac{\lambda_{t,s_j^t}}{C}-\sum_i \alpha_{ij}^t k(x_t,\cdot)=0,$$

where $(\alpha_{ij}^t)\in\mathcal{A}_t, (r_i^t\times s_j^t)\in\mathcal{B}_t$, $\mathcal{A}_t=[A|A\in R_+^{\|Y_t\|}\times R_+^{(k-\|Y_t\|)},\|A\|_1\le 1]$ and $\mathcal{B}_t=Y_t\times([k]/Y_t)$. Furthermore, we set $\alpha_{ij}$ to zeros for $\forall(i,j)\in\{[Y_t\times([k]/Y_t)]/(r_t,s_t)\}$. For simplicity, we denote $\alpha_{r_t,s_t}^t$ as $\frac{\gamma_t}{C}$. As a result, the dual objective function becomes

$$D(\gamma_1,\cdots,\gamma_T)=-\sum_{i=1}^k\frac{1}{2}\|\sum_{t=1}^T\sigma(i,t)\gamma_t\kappa(x_t,\cdot)\|^2+\sum_{t=1}^T\gamma_t,$$

where $\gamma_t\in[0,C]$ and $\sigma(i,t)=\begin{cases}1 & \text{if } i=r_t\\ -1 & \text{if } i=s_t\\ 0 & \text{otherwise}\end{cases}$. ■

By applying Lemma 3, we thus have the dual objective function for the $t$-th step as:

$$D_t(\gamma_1,\cdots,\gamma_t)=-\sum_{i=1}^k\frac{1}{2}\|\sum_{j=1}^t\sigma(i,j)\gamma_j k(x_j,\cdot)\|^2+\sum_{j=1}^t\gamma_j. \tag{10}$$

Now our goal is to derive the dual ascent guaranteed by the proposed double updating scheme. When pair $(x_a,Y_a)$ is misclassified by the prediction function $\overline{f_t}=(f_{t,1},\cdots,f_{t,k})$, we will perform the update on the prediction function. Assume we conduct a double updating for $(x_a,Y_a)$ and some auxiliary example $(x_b,Y_b)$, we can prove Proposition 4 as follows.

**Proof** According to Equation (10) obtained by Lemma 3, before performing the double updating, the value of the dual function is expressed as:

$$D_{t-1}=-\sum_{i=1}^k\frac{1}{2}\|\sum_{j=1}^{t-1}\sigma(i,j)\hat{\gamma}_j k(x_j,\cdot)\|^2+\sum_{j=1}^{t-1}\hat{\gamma}_j=-\sum_{i=1}^k\frac{1}{2}\|f_{t-1,i}\|^2+\sum_{j=1}^{t-1}\hat{\gamma}_j,$$

where $\hat{\gamma}_j$'s denote the weights of the prediction function $\overline{f_{t-1}}$ before the updating. After performing the dual update, the value of the new dual function can be written as:

$$D_t=-\sum_{i=1}^k\frac{1}{2}\|f_{t-1,i}+\sigma(i,a)\gamma_a k(x_a,\cdot)+\sigma(i,b)d_{\gamma_b}k(x_b,\cdot)\|^2+\sum_{j=1}^{t-1}\hat{\gamma}_j+\gamma_a+d_{\gamma_b}.$$

Hence, the dual ascent is computed as follows:

$$\begin{aligned}\Delta D=D_t-D_{t-1} \quad = \quad & \gamma_a\Big(1-\big(f_{t-1,r_a}(x_a)-f_{t-1,s_a}(x_a)\big)\Big)+d_{\gamma_b}\Big(1-\big(f_{t-1,r_b}(x_b)-f_{t-1,s_b}(x_b)\big)\Big)\\ & -\gamma_a^2 s_a-d_{\gamma_b}^2 s_b-\sum_{i=1}^k\sigma(i,a)\sigma(i,b)\gamma_a d_{\gamma_b}k(x_a,x_b)\ .\end{aligned}$$

■

## References

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with larank. In *Proceedings of the 24th International Conference on Machine learning (ICML'07)*, pages 89–96, 2007.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 409–415, 2000.

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

Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Inf. Theory*, 50(9):2050–2057, 2004.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

Koby Crammer and Yoram Singer. Loss bounds for online category ranking. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT'05)*, pages 48–62, 2005.

Koby Crammer, Jaz S. Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Koby Crammer, Mark Dredze, and Fernando Pereira. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 345–352, 2008.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008. ISSN 0097-5397.

Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*, pages 264–271, 2008.

Michael Fink, Shai Shalev-Shwartz, Yoram Singer, and Shimon Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 25th International Conference on Machine learning (ICML'06)*, pages 313–320, 2006.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, 1999.

Claudio Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

Jyrki Kivinen and Manfred K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 209–218, 1995.

Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *Advances in Neural Information Processing Systems (NIPS)*, pages 785–792, 2001.

Yi Li and Philip M. Long. The relaxed online maximum margin algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 498–504, 1999.

Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML'08)*, pages 720–727, 2008.

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

Shai Shalev-Shwartz. Online learning:theory, algorithms, and applications. In *Ph.D thesis*, 2007.

Shai Shalev-Shwartz and Yoram Singer. Online learning meets optimization in the dual. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT'06)*, pages 423–437, 2006.

Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Jason Weston and Antoine Bordes. Online (and offline) on an even tighter budget. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS'05)*, pages 413–420, 2005.

Peilin Zhao, Steven C. H. Hoi, and Rong Jin. Duol: A double updating approach for online learning. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 2259–2267, 2009.