A better analysis of a Perceptron-based algorithm for multiclass importance weighted online learning

Adisak Supeesun, Jittat Fakcharoenphol
Department of Computer Engineering,
Kasetsart University,
Bangkok, 10900 Thailand.
E-mails: adisak.sup@gmail.com, jittat@gmail.com

Abstract

This paper considers an online learning problem with importance weighted examples. We present two results.

The first result, concerning binary classification, is an improved bound on the weighted mistakes for a simple algorithm based on the Perceptron algorithm. The previous bound uses the result of Gentile [7] on the p-norm algorithm in a simple way and has a multiplicative dependence on the weight upon the loss bound. This paper gets rid of that dependence by a careful modification of the proof of Gentile.

Using the first result as a black-box, the general online multiclass framework of Fink, Shalev-Shwartz, Singer, and Ullman [6] can be shown to work in the importance weighted setting.

Our second result is related to the general idea of multiclass-binary learning reductions. In this online settings, we show that two different methods, the multi-vector method and its particular all-pair variant, for reducing multiclass problems to binary problems are equivalent.

1 Introduction

Machine learning has been a key factor in numerous successful computing applications. The usual approach starts from training example collection, then the classifiers are trained, and finally they are applied in the real environment. The quality of the constructed classifiers depends on many factors; one of them is that the training examples should represent the real testing data. In many learning tasks, this requirement is not difficult to achieve, however, in many situations, it is hard or impossible to do so, for examples, in the case of changing environment or the case where examples are expensive.

One approach to remedy the problem of finding good training examples is to avoid having to train the classifiers before their use, but allowing them to learn while in their operation. Informally, this setting is called *online learning*, in contrast to the usual approach of *offline learning* or *batch learning*. (For general introduction to online learning, see [2].)

In this paper we consider online learning in the case that examples are weighted. The classifier receives each example, predicts its label, and learns of the real label and the weight of the example. Its penalty (or cost) depends on the weights of examples.

In an offline setting, this problem has received quite a considerable attention [4, 13, 12].

In our previous work, we have shown that a simple modification of the Perceptron algorithm [10] works well for this problem, when the number of classes is two, i.e., the binary classification problem. We show that the algorithm performance is comparable to *any* fixed perceptrons. However, our bounds depend heavily on the largest weight of the examples. (See Theorem 2.)

This paper improves the bound so that our algorithm is "competitive" with some additive constant against any fixed perceptrons, while our previous bound contains the largest weight as an extra multiplicative factor. (See theorem 3.) Our proof generalizes the proof of the perceptron performance of Gentile [7] in a straight-forward way to include the case of weighted examples.

For online multiclass problems, Fink, Shalev-Shwartz, Singer, and Ullman [6] give a framework for combining two main methods for online multiclass learning, i.e., the single-vector method and the multi-vector method. Their performance proof use the result of Gentile [7].

Therefore, our generalization of Gentile's result implies that the result of Fink *et al.* carries over to the importance weighted settings as well.

Another contribution of this paper is to show the equivalence between the multi-vector method and another method using an all-pair approach (e.g., like the Max-win algorithm) with linear scoring.

In the multi-vector method using the perceptron algorithm, each class maintains a weight vector that distinguishes itself from other classes. (A more detail description is provided in Section 2.3.) A natural extension to this algorithm that we consider maintains a weight vector for each pair of classes i and j that is used to distinguishes between instances from class i and class j. In this paper, we show that the all-pair method is equivalent to the multi-vector method, i.e., if both algorithms start with equivalent weights vectors, including the case that they start with zero weight vectors, they would always produce the same predic-

tions. This is true in both weighted and unweighted cases.

The comparison between these two algorithms can be placed into bigger debates in Machine Learning. We note that these two approaches are to one another like the One-against-All and the One-against-One approaches in the off-line multiclass learning setting. It remains one of the important open problems to understand both reduction methods better. (See [9].)

1.1 Organization

In Section 2, we define the problems and discuss previous work. We summarize our contribution in Section 2.4. Section 3 proves an improved bound for binary learning, while Section 4 deals with multiclass learning. We describe an all-pair method for on-line multiclass learning in Section 4.1 and prove its equivalence to the multi-vector method in Section 4.2. Finally, Section 5 shows preliminary experimental results.

2 Problem setting

Online learning proceeds in rounds. For round t, the classifier receives an instance $\mathbf{x}_t \in \mathbb{R}^n$. It has to predict the label $\hat{y}_t$ of $\mathbf{x}_t$. After the prediction is done, the true label y_t of $\mathbf{x}_t$ is revealed. We denote the set of possible labels by $\mathcal{Y}$. In many cases, $\mathcal{Y}$ is known, e.g., in binary learning problem $\mathcal{Y} = \{+1, -1\}$. However, in other cases, the set $\mathcal{Y}$ is not known in advanced; thus, we let $\mathcal{Y}_t$ denote the set of all labels the algorithm have seen so far before round t. We say that the algorithm makes a mistake if $\hat{y}_t \neq y_t$ and $y_t \in \mathcal{Y}_t$. While Fink *et al*. use the latter settings, for simplicity, we assume that we know all the classes before hand.

If the examples are unweighted, we want the algorithm to minimize the number of mistakes. We are interested in the case of weighted examples. Together with the true label y_t, the algorithm receives the cost $c_t \in [0, \infty)$; for any round t that the algorithm makes a mistake the cost of c_t is incurred. The goal is to minimize the total cost. Let c_{max} and c_{min} denote the maximum and the minimum cost, respectively. These two constants are used in our analysis only; the algorithm does not need them. Also let θ denote the ratio $\frac{c_{max}}{c_{min}}$.

Next in this section, we give definitions regarding loss for binary classification and multiclass classification.

2.1 Binary classification

In round t, the prediction is done by a hypothesis h_t. For binary classification h_t is a function from $\mathbb{R}^n$ to $\mathbb{R}$. The algorithm would predict $\mathrm{sgn}(h_t(\mathbf{x}_t))$, where $\mathrm{sgn}(\cdot)$ is a sign function. To measure the performance of the algorithm, we use the *hinge loss* function, defined as

$$\ell_t(h) = (1 - y_t h(\mathbf{x}_t))_+ ,$$

where $(a)_+ = \max\{a, 0\}$. Note that when the algorithm make a mistake the its hinge loss is at least 1; thus hinge loss function is an upperbound on the mistake.

For example weighted binary classification, we extend the notion of hinge loss to include weights, i.e., *weighted hinged loss* for h in round t is defined as

$$\ell_t(h) = c_t \left(1 - y_t h(\mathbf{x}_t)\right)_+ .$$

Note that for brevity, we use the same notation for unweighted and weighted hinge loss. Their uses shall be clear from the context.

We measure the performance of the algorithm with its *cumulative hinge loss*.

In this work, we use perceptrons as classifiers, i.e., our hypothesis h_t is a linear function $h_t(x_t) = \langle \mathbf{w}_t, \mathbf{x}_t \rangle$, for some $\mathbf{w}_t \in \mathbb{R}^n$.

2.2 Multiclass classification

We follow the definition in Fink *et al* [6]. The hypothesis h_t for round t is a function $h_t : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}$, which gives the score for each class $y \in \mathcal{Y}$. The algorithm predicts label $\hat{y}_t = \arg\max_{y \in \mathcal{Y}_t} h_t(y, \mathbf{x}_t)$. We define the *hinge loss* to be

$$\ell_t(h) = \left(1 - h(\mathbf{x}_t, y_t) + \max_{r \in \mathcal{Y}_t \setminus \{y_t\}} h(\mathbf{x}_t, r)\right)_+ .$$

Note the the hinge loss measures the difference between the score of the correct class y_t and the score of the predicted class $\hat{y}_t$.

As in the binary problem, we can define the *weighted hinge loss* as

$$\ell_t(h) = c_t \left(1 - h(\mathbf{x}_t, y_t) + \max_{r \in \mathcal{Y}_t \setminus \{y_t\}} h(\mathbf{x}_t, r)\right)_+ ,$$

when $y_t \in \mathcal{Y}_t$.

2.3 Previous work

The mistake bound [8] of the Perceptron algorithm when the data from the two classes are linearly separable is a classic result in machine learning. The general case which has been recently proved by Gentile [7] forms the basis of the work of Fink *et al* and the current work. Gentile show the relation of the number of mistakes the Perceptron algorithm makes and the hinge loss of any linear classifier. The simpler form of the theorem by Gentile, as stated in [6] is as follows.

Theorem 1 ([7], as stated in[6]) *For any input sequence* $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^n \times \{-1, +1\})^m$, *define* $R = \max_t \|\mathbf{a}_t\|_2$. *If there exist a linear classifier* $h^\star$ *defined by weight vector* $\mathbf{u}$ *where* $\sum_{t=1}^m \ell_t(h^\star) \leq L$ *and define* $C = R^2 \|\mathbf{u}\|_2^2$, *then the mistakes on* S *of the perceptron algorithm is at most* $L + C + \sqrt{LC}$.

In our previous paper [11], we modify the original Perceptron algorithm to handle weights as follows. We start with weight $\mathbf{w}_1 = 0$, let $h_t(\mathbf{x}_t) = \langle \mathbf{w}_t, \mathbf{x}_t \rangle$, and let $\hat{y}_t = \mathrm{sgn}(h_t(\mathbf{x}_t))$. For each round t the algorithm makes a mistake, it updates the weight using the following operation

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + c_t y_t \mathbf{x}_t,$$

i.e., the algorithm simply updates the weight vector with $\mathbf{x}_t$, the instance, scaled according to its importance weight. Later, we shall call this algorithm as Importance Weighted Perceptron or IWP.

We use a more general form of the Theorem 1 to prove the following loss bound.

Theorem 2 ([11]) *For any weighted input sequence $S = ((\mathbf{x}_1, y_1, c_1), \ldots, (\mathbf{x}_m, y_m, c_m)) \in (\mathbb{R}^n \times \{-1, +1\}, [0, \infty))^m$, define $R = \max_t \|\mathbf{x}_t\|_2$. If there exists a linear classifier $h^\star$ defined by weight vector $\mathbf{u}$ such that $\sum_{t=1}^m \ell_t(h^\star) \le L$ and define $C = R^2\|\mathbf{u}\|_2^2$ and $c_{max} = \max c_t$, then the weighted loss on S of the IWP algorithm is at most*

$$\theta L + c_{max}\theta^2 C + \theta^2\sqrt{c_{min} LC},$$

where θ is the ratio $\frac{c_{max}}{c_{min}}$.

We note the dependence on the ratio θ.

For multiclass online learning, Fink, Shalev-Shwartz, Singer, and Ullman [6] consider two methods for dealing with on-line multiclass learning: the multi-vector method and the single-vector method. They present an general mixing framework that includes both methods and obtained a mistake bound using the analysis of Gentile.

Here we review only the multi-vector method, which is a standard approach for dealing with multiclass learning (see, e.g., [5]). In this approach, each class r maintains a vector $\mathbf{w}_t^r$ for round t. For a given input $\mathbf{x}$, it computes the score for class r as

$$h_t(\mathbf{x}, r) = \langle \mathbf{w}_t^r, \mathbf{x} \rangle,$$

and predict the class $\hat{y}_t = \arg\max_{r \in \mathcal{Y}} h(\mathbf{x}, r)$. If the algorithm makes a mistake, i.e., the correct class is $y_t \neq \hat{y}_t$, it updates the weights for class y_t and $\hat{y}_t$, i.e.,

$$\mathbf{w}_{t+1}^{y_t} = \mathbf{w}_t^{y_t} + \mathbf{x}_t, \quad \mathbf{w}_{t+1}^{\hat{y}_t} = \mathbf{w}_t^{\hat{y}_t} - \mathbf{x}_t,$$

and let $\mathbf{w}_{t+1}^r = \mathbf{w}_t^r$ for $r \notin \{y_t, \hat{y}_t\}$.

The paper of Crammer, Dekel, Keshet, Shalev-Shwartz and Singer [3] also consider the problem of multiclass online learning when example have weights. However, the weights depends on the class of the examples, and examples from the same class have the same weights. While our result works when weights depend on each example.

2.4 Our results

This paper presents two main results.

1. Improved bound on Theorem 2. In this paper, we consider the proof of Gentile [7] in details, and generalize it so that it includes the case of weighted examples. We are able to obtain the following theorem.

Theorem 3 *For any weighted input sequence $S = ((\mathbf{x}_1, y_1, c_1), \ldots, (\mathbf{x}_m, y_m, c_m)) \in (\mathbb{R}^n \times \{-1, +1\}, [0, \infty))^m$, define $R = \max_t \|\mathbf{x}_t\|_2$. If there exists a linear classifier $h^\star$ defined by weight vector $\mathbf{u}$ such that $\sum_{t=1}^m \ell_t(h^\star) \le L$ and define $C = R^2\|\mathbf{u}\|_2^2$ and $c_{max} = \max c_t$, then the weighted loss on S of the IWP algorithm is at most*

$$L + c_{max} C + \sqrt{c_{max} LC}.$$

We note a better dependency at the L term in the above theorem.

2. Prove the equivalence between an natural all-pair method with linear scoring and the multi-vector method. As noted in the introduction, this pair of methods are to each other as the One-against-One and One-against-All approaches in the off-line multiclass learning.

3 Importance weighted binary classification

In this section, we prove Theorem 3. In fact, we shall prove a more general theorem, and Theorem 3 will be its corollary. Our proof follows the proof of Gentile [7] for the p-norm algorithm.

We start with definitions. For any vector $\mathbf{u}$ and margin $\gamma > 0$, the weighted deviation $D_\gamma(\mathbf{u}; (\mathbf{x}, y, c))$ is defined as

$$D_\gamma(\mathbf{u}; (\mathbf{x}, y, c)) = c(\gamma - y\langle \mathbf{u}, \mathbf{x} \rangle)_+.$$

Let the sequence

$$S = ((\mathbf{x}_1, y_1, c_1), \ldots, (\mathbf{x}_m, y_m, c_m)),$$

be the weighted input sequences, where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{-1, +1\}$, and $c_t \in [0, \infty)$.

Let M denote the set of rounds that the IWP makes mistake. The weighted loss $Loss(S)$ over the input sequence S is defined as $\sum_{t \in M} c_t$.

To measure the distance between vectors $\mathbf{u}$ and $\mathbf{w}$, we use the following function

$$d(\mathbf{u}, \mathbf{w}) = \frac{1}{2}\|\mathbf{u}\|_2^2 + \frac{1}{2}\|\mathbf{w}\|_2^2 - \langle \mathbf{u}, \mathbf{w} \rangle.$$

This is a specific form of the Bregman divergence when $p = 2$ used in Gentile [7].

We shall use the following two lemmas, proved in a more general setting in Gentile [7] (as Lemma 6 and Lemma 4).

Lemma 1 *Let $\mathbf{u}, \mathbf{w}, \mathbf{x} \in \mathbb{R}^n$, $a \in \mathbb{R}$ and set $\mathbf{w}' = \mathbf{w} + a\mathbf{x}$. The following equality holds:*

$$a(\mathbf{u} \cdot \mathbf{x} - \mathbf{w} \cdot \mathbf{x}) = d(\mathbf{u}, \mathbf{w}) - d(\mathbf{u}, \mathbf{w}') + d(\mathbf{w}, \mathbf{w}').$$

Lemma 2 *Let $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$, $a \in \mathbb{R}$ and set $\mathbf{w}' = \mathbf{w} + a\mathbf{x}$. We have*

$$d(\mathbf{w}, \mathbf{w}') \le \frac{a^2}{2}\|\mathbf{x}\|_2^2.$$

We state the theorem below. Its proof is very technical and is quite straight-forward modification of Theorem 8 in Gentile [7]. Therefore, in this version of the paper, we only sketch the modification. In the final version, we will give the full proof.

Theorem 4 *For any weighted input sequence* $S = ((\mathbf{x}_1, y_1, c_1), \ldots, (\mathbf{x}_m, y_m, c_m)) \in (\mathbb{R}^n \times \{-1, +1\}, [0, \infty))^m$*, let* M *be a set of trial that the modified Perceptron algorithm makes a mistake when it is run on* S *and* $\mathbf{w}_1 \in \mathbb{R}^n$*, and define* $R = max_{t \in M} \|\mathbf{x}_t\|_2$*. For any* $\mathbf{u} \in \mathbb{R}^n$ *the modified Perceptron algorithm achieves the weighted loss*

$$\begin{aligned} Loss(S) &\leq \frac{1}{\gamma} L_{\mathbf{u},\gamma}(M) - \frac{c_{max}\langle \mathbf{u}, \mathbf{w}_1 \rangle}{\gamma} + \frac{c_{max} C}{2\gamma^2} \\ &\quad + \frac{c_{max}}{2\gamma^2} \sqrt{\frac{4\gamma C L_{\mathbf{u},\gamma}(M)}{c_{max}} + Z} \end{aligned}$$

where $L_{\mathbf{u},\gamma}(M) = \sum_{t \in M} D_\gamma(\mathbf{u}; (\mathbf{x}_t, y_t, c_t))$*,* $C = R^2 \|\mathbf{u}\|_2^2$*,* $\gamma > 0$*,* $c_{max} = \max_{t \in M} c_t$*, and* $Z = C^2 - 4\gamma C \langle \mathbf{u}, \mathbf{w} \rangle + 4\gamma^2 \|\mathbf{u}\|_2^2 \|\mathbf{w}_1\|_2^2$*.*

Proof: (sketch) For simplicity, we first assume that $c_t \leq 1$, for all $1 \leq t \leq m$.

Since the term in the radical sign is non-negative, if $Loss(S)$ is at most $\frac{1}{\gamma} L_{\mathbf{u},\gamma}(M) - \frac{c_{max}\langle \mathbf{u}, \mathbf{w}_1 \rangle}{\gamma}$ the theorem holds. Therefore, we now assume that $Loss(S) > \frac{1}{\gamma} L_{\mathbf{u},\gamma}(M) - \frac{c_{max}\langle \mathbf{u}, \mathbf{w}_1 \rangle}{\gamma}$.

We consider the round $t \in M$, where the weight changes. Using lemma 1 (setting $a = c_t y_t$, $\mathbf{u}$ with $\lambda \mathbf{u}$), we have that

$$\lambda y_t \mathbf{u} c_t \mathbf{x}_t \leq d(\lambda \mathbf{u}, \mathbf{w}_t) - d(\lambda \mathbf{u}, \mathbf{w}_{t+1}) + d(\mathbf{w}_t, \mathbf{w}_{t+1}).$$

Let $\hat{\gamma}_{\mathbf{u},M} = \frac{1}{Loss(S)} \sum_{t \in M} y_t c_t \mathbf{x}_t$. By summing the previous equation over $t \in M$, we obtain:

$$\begin{aligned} \lambda \hat{\gamma}_{\mathbf{u},M} Loss(S) &\leq d(\lambda \mathbf{u}, \mathbf{w}_1) - d(\lambda \mathbf{u}, \mathbf{w}_{m+1}) \\ &\quad + \sum_{t \in M} d(\mathbf{w}_t, \mathbf{w}_{t+1}) \end{aligned}$$

We bound the last term with Lemma 2 (setting $a = y_t c_t$); i.e., we have

$$\sum_{t \in M} d(\mathbf{w}_t, \mathbf{w}_{t+1}) \leq \sum_{t \in M} \frac{c_t^2}{2} \|\mathbf{x}_t\|_2^2.$$

However, since $c_t \leq 1$, $c_t^2 \leq c_t$. Hence, the last term can be bounded as follows

$$\begin{aligned} \sum_{t \in M} \frac{c_t}{2} \|\mathbf{x}_t\|_2^2 &\leq \sum_{t \in M} \frac{c_t}{2} R^2 \\ &= \frac{R^2}{2} \sum_{t \in M} c_t \\ &= \frac{R^2}{2} Loss(S). \end{aligned}$$

From here, we can follow Gentile's proof to show the relation between weighted deviation of $\mathbf{u}$ and $\hat{\gamma}_{\mathbf{u},M}$, and upper bound $Loss(S)$ as

$$\begin{aligned} Loss(S) &\leq \frac{1}{\gamma} L_{\mathbf{u},\gamma}(M) - \frac{\langle \mathbf{u}, \mathbf{w}_1 \rangle}{\gamma} + \frac{C}{2\gamma^2} \\ &\quad + \frac{1}{2\gamma^2} \sqrt{4\gamma C L_{\mathbf{u},\gamma}(M) + C^2 + Z} \end{aligned}$$

To see the general case, i.e., when the assumption $c_{max} \leq 1$ is taken off, we note that we can multiply the above bound by c_{max} and get the claimed bound. ∎

We note that by some manipulation of Theorem 4, Theorem 3 can be obtained. We omit the detail in this version.

4 Importance weighted multi-class classification

The proof of in Fink *et al* [6] uses the analysis of the Perceptron algorithm of Gentile [7]. Since Theorem 3 generalizes that analysis to the case of importance weighted examples, Fink *et al* algorithm can also be used in the weighted settings as well.

The framework of Fink *et al* unifies the single-vector and multi-vector methods. We note that in the offline multiclass learning literature, the multi-vector method belongs to the class of One-Against-All method for reducing multiclass classification to binary classification.

In Section 4.1, motivated by the One-against-One method (see, e.g., [9]), we define another method for online multiclass learning using linear classifiers called the all-pair method. We sketch the proof that it is equivalent to the multi-vector method in Section 4.2.

4.1 The all-pair method

We assume that the set $\mathcal{Y}$ is known. With out loss of generality we assume that $\mathcal{Y} = \{1, 2, \ldots, k\}$, i.e., we have k classes. We maintain a set of $k \cdot (k-1)$ weights, i.e., for each pair i and j from $\mathcal{Y}$, we have $\mathbf{w}_t^{ij}$ for round t. For simplicity, we also assume $\mathbf{w}_t^{ij} = -\mathbf{w}_t^{ji}$, i.e., the weights are anti-symmetric. Intuitively, we want, for each instance $\mathbf{x}$ in class i, $\langle \mathbf{w}_t^{ij}, \mathbf{x} \rangle \geq 1$, for every $j \neq i$.

Given the set of weights, we define the score function

$$h_t(\mathbf{x}_t, r) = \sum_{j \in \mathcal{Y} \setminus \{r\}} \langle \mathbf{w}_t^{rj}, \mathbf{x}_t \rangle,$$

i.e., the score of class r equals the sum of the scores class r receives from each binary classifiers.

We note that although this is similar to Max-Win, the scores we consider here is linear, and, in Max-Win terminology, if class r loses to some other class j, the total score of r is penalized also by that loss as well.

After the prediction, the algorithm receives the true class y_t and the importance weight c_t. If the algorithm

makes a mistake, i.e., $\hat{y}_t \neq y_t$, it updates the weights that contribute to the score of class y_t and $\hat{y}_t$. I.e., for each $r \neq y_t$,

$$\mathbf{w}_{t+1}^{y_t r} \leftarrow \mathbf{w}_t^{y_t r} + c_t \cdot \mathbf{x}_t,$$

Then, for each $r \neq \hat{y}_t$,

$$\mathbf{w}_{t+1}^{\hat{y}_t r} \leftarrow \mathbf{w}_t^{\hat{y}_t r} - c_t \cdot \mathbf{x}_t.$$

Note that these updates affect $\mathbf{w}_{t+1}^{r y_t}$ and $\mathbf{w}_{t+1}^{r \hat{y}_t}$ as well, since the weights are anti-symmetric. Also, weight vector $\mathbf{w}_{t+1}^{y_t \hat{y}_t}$ is updated twice.

We note that other scoring schemes are known, e.g., the zero-one scoring scheme of Max-Win. We do not know how to analyze them. Empirically, they produce roughly the same performances. In the final version we shall include this comparison in the experiment section.

4.2 The equivalence

For simplicity, we prove only the unweighted case, i.e., $c_t = 1$, for every round t; the importance weighted case is similar. We recall the notations for the multi-vector method from Section 2.3.

We shall refer to the multi-vector method as MV and the all-pair method as AP. We also that both algorithms break ties in the same way.

For round t, we let $h_t^M(\mathbf{x}, r)$ denote the score of class r for x in MV and $h_t^A(\mathbf{x}, r)$ denote the score in AP.

We start with a simple observation. If, for round t, $h_t^A(\mathbf{x}, r) = k \cdot h_t^M(\mathbf{x}, r)$ for any $\mathbf{x} \in R^n$ and for any class r, both MV and AP give the same prediction for every input $\mathbf{x}$. This is because if $h_t^M(\mathbf{x}, i) > h_t^M(\mathbf{x}, j)$, $h_t^A(\mathbf{x}, i) > h_t^A(\mathbf{x}, j)$.

The next lemma is the key.

Lemma 3 *If, for round t, $h_t^M(\mathbf{x}, r) = k \cdot h_t^A(\mathbf{x}, r)$ for any $x \in R^n$ and r, in the next round, we have*

$$h_{t+1}^A(\mathbf{x}, r) = k \cdot h_{t+1}^M(\mathbf{x}, r),$$

for any $x \in \mathbb{R}^n$ and r.

Proof: Note that if they both give correct prediction in round t, we do not update weights, and therefore the lemma remains true.

Assume that they make a mistake in round t for input $\mathbf{x}_t$. Since they give the same prediction in round t, assume that they predict class $\hat{y}_t$ while the correct class is class y_t.

We first consider how MV updates weights. Note that all weights $\mathbf{w}_t^r$ for $r \notin \{y_t, \hat{y}_t\}$ remain unchanged. It only updates

$$\mathbf{w}_{t+1}^{y_t} = \mathbf{w}_t^{y_t} + \mathbf{x}_t, \quad \mathbf{w}_{t+1}^{\hat{y}_t} = \mathbf{w}_t^{\hat{y}_t} - \mathbf{x}_t.$$

Therefore, for any input $\mathbf{x}$,

$$\begin{aligned} h_{t+1}^M(\mathbf{x}, y_t) &= \langle \mathbf{w}_t^{y_t} + \mathbf{x}_t, \mathbf{x} \rangle = \langle \mathbf{w}_t^{y_t}, \mathbf{x} \rangle + \langle \mathbf{x}_t, \mathbf{x} \rangle \\ &= h_t^M(\mathbf{x}, y_t) + \langle \mathbf{x}_t, \mathbf{x} \rangle. \end{aligned} \tag{1}$$

Similarly, we have

$$h_{t+1}^M(\mathbf{x}, \hat{y}_t) = h_t^M(\mathbf{x}, \hat{y}_t) - \langle \mathbf{x}_t, \mathbf{x} \rangle. \tag{2}$$

We analyze the score of AP for an input $\mathbf{x}$. First, consider any class $r \notin \{y_t, \hat{y}_t\}$. Note that

$$\begin{aligned} h_{t+1}^A(\mathbf{x}, r) &= \textstyle\sum_{j \in \mathcal{Y} \setminus \{r\}} \langle \mathbf{w}_{t+1}^{rj}, \mathbf{x} \rangle, \\ &= \left\langle \textstyle\sum_{j \in \mathcal{Y} \setminus \{r\}} \mathbf{w}_{t+1}^{rj}, \mathbf{x} \right\rangle, \end{aligned}$$

by the property of inner products. Recall that among all the weights involved the algorithm only updates $\mathbf{w}_{t+1}^{r y_t}$ and $\mathbf{w}_{t+1}^{r \hat{y}_t}$, i.e.,

$$\mathbf{w}_{t+1}^{r y_t} = \mathbf{w}_t^{r y_t} - \mathbf{x}_t, \quad \mathbf{w}_{t+1}^{r \hat{y}_t} = \mathbf{w}_t^{r \hat{y}_t} + \mathbf{x}_t,$$

which cancel out in the sum of weights, i.e.,

$$\textstyle\sum_{j \in \mathcal{Y} \setminus \{r\}} \mathbf{w}_{t+1}^{rj} = \sum_{j \in \mathcal{Y} \setminus \{r\}} \mathbf{w}_t^{rj}.$$

Thus,

$$h_{t+1}^A(\mathbf{x}, r) = h_t^A(\mathbf{x}, r), \tag{3}$$

for any x and $r \in \mathcal{Y} - \{y_t, \hat{y}_t\}$.

Consider $h_{t+1}^A(\mathbf{x}, y_t) = \sum_{j \in \mathcal{Y} \setminus \{y_t\}} \langle \mathbf{w}_{t+1}^{y_t j}, \mathbf{x} \rangle$. Note that for each $r \notin \{y_t, \hat{y}_t\}$,

$$\mathbf{w}_{t+1}^{y_t r} \leftarrow \mathbf{w}_t^{y_t r} + \mathbf{x}_t,$$

and for class $\hat{y}_t$, we update $w^{y_t \hat{y}_t}$ twice, i.e.,

$$\mathbf{w}_{t+1}^{y_t \hat{y}_t} \leftarrow \mathbf{w}_t^{y_t \hat{y}_t} + 2 \cdot \mathbf{x}_t.$$

Thus,

$$\textstyle\sum_{j \in \mathcal{Y} \setminus \{y_t\}} \mathbf{w}_{t+1}^{y_t j} = \sum_{j \in \mathcal{Y} \setminus \{y_t\}} \mathbf{w}_t^{y_t j} + k \cdot \mathbf{x}_t.$$

Plugging in, we have

$$\begin{aligned} h_{t+1}^A(\mathbf{x}, y_t) &= \left\langle \left(\textstyle\sum_{j \in \mathcal{Y} \setminus \{y_t\}} \mathbf{w}_t^{y_t j} + k \cdot \mathbf{x}_t \right), \mathbf{x} \right\rangle \\ &= \left\langle \textstyle\sum_{j \in \mathcal{Y} \setminus \{y_t\}} \mathbf{w}_t^{y_t j}, \mathbf{x} \right\rangle + k \cdot \langle \mathbf{x}_t, \mathbf{x} \rangle \\ &= h_t^A(\mathbf{x}, y_t) + k \cdot \langle \mathbf{x}_t, \mathbf{x} \rangle. \end{aligned} \tag{4}$$

Similarly,

$$h_{t+1}^A(\mathbf{x}, \hat{y}_t) = h_t^A(\mathbf{x}, \hat{y}_t) - k \cdot \langle \mathbf{x}_t, \mathbf{x} \rangle. \tag{5}$$

The lemma follows from equations (1)–(5). ∎

Note that the assumption of this lemma is true for $t = 0$ if all weights are initially zero. From this lemma, it is not difficult to prove the required equivalence; we omit the proof due to space limitation.

Table 1: The table below shows the average number of mistakes and average weighted mistakes for each algorithm over 10 runs on the weighted input sequence from LETTER, ISOLET, CAR and ABALONE data sets.

Data set	MV-IWP		MV-Perceptron	
	mistake	cost	mistake	cost
LETTER	64.22	**51.33**	**55.82**	57.29
ISOLET	40.74	**23.75**	**25.85**	26.51
CAR	32.17	**38.01**	**20.56**	44.01
ABALONE	93.32	95.12	**84.94**	**93.96**

5 Experiments

The comparison of IWP and the standard Perceptron algorithm on importance weighted examples has been show briefly in our previous paper [11]. In this paper, we focus more on multiclass problems. We use the multi-vector method with the Perceptron and the IWP. We conduct experiments on other standard data sets in UCI Machine Learning Repository [1].

We consider a four data sets: LETTER, ISOLET, CAR and ABALONE. There are 20000 instances, each with 16 attributes in LETTER, 6238 instances, each with 617 attributes in ISOLET, 1728 instances, each with 6 attributes in CAR and 4177 instances each with 8 attributes in ABALONE. LETTER and ISOLET have 26 classes, CAR has 4 classes, and ABALONE has 28 classes.

As in [9], we convert each nominal attribute with b possible values in CAR and ABALONE to b binary attributes.

We generate an importance weight of each example for the LETTER and ISOLET according to its true label, i.e., the weight of each example is inversely proportional to its frequency of its corresponding English character in use. For the CAR and ABALONE data sets, We generate the importance weight of each example in class $y \in \mathcal{Y}$ to be $\frac{1}{P(y)}$, where $P(y)$ be the probability of occurrence of class y in the data sets.

For each run, we permute the examples in the data sets, and run them through 2 algorithms: multi-vector algorithms with standard Perceptron and with the IWP algorithm. For each data set, we conduct 10 experiments. Table 1 shows the number of average mistakes (over the number of rounds) and the average weighted cost (over the total cost). The total cost of the LETTER data set is 77011, 23995.3 for the ISOLET, 6911.85 for the CAR, and 116956 for the ABALONE.

From the table, we can see that the algorithm based on Perceptron algorithm has the minimum number of mistakes. However, the one with IWP has the minimum total weighted cost in three data sets.

We note that the ABALONE data set is a hard one. Even good off-line algorithms for unweighted version have mistake rates of at least 70% (see, e.g., [9]).

References

[1] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.

[2] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[3] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, 2006.

[4] P. Domingos. How to get a free lunch: A simple cost model for machine learning applications, 1998.

[5] Duda and Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[6] Michael Fink, Shai Shalev-Shwartz, Yoram Singer, and Shimon Ullman. Online multiclass learning by interclass hypothesis sharing. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 313–320, New York, NY, USA, 2006. ACM.

[7] Claudio Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

[8] Albert B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

[9] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, 2004.

[10] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, November 1958.

[11] Adisak Supeesun and Jittat Fakcharoenphol. Online learning for importance weighted classification (in thai). In *NCSEC'2007*, pages 798–805, 2007. Also available at http://www.cpe.ku.ac.th/˜jtf/research.html.

[12] Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–213, New York, NY, USA, 2001. ACM.

[13] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 435, Washington, DC, USA, 2003. IEEE Computer Society.