Journal of Machine Learning Research 3 (2003) 951-991 Submitted 2/02; Published 1/03

Ultraconservative Online Algorithms for Multiclass Problems

Koby Crammer KOBICS@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science & Engineering
Hebrew University, Jerusalem 91904, Israel

Editor: Manfred K. Warmuth

Abstract

In this paper we study a paradigm to generalize online classification algorithms for binary classification problems to multiclass problems. The particular hypotheses we investigate maintain one prototype vector per class. Given an input instance, a multiclass hypothesis computes a similarity-score between each prototype and the input instance and sets the predicted label to be the index of the prototype achieving the highest similarity. To design and analyze the learning algorithms in this paper we introduce the notion of *ultraconservativeness*. Ultraconservative algorithms are algorithms that update only the prototypes attaining similarity-scores which are higher than the score of the correct label's prototype. We start by describing a family of additive ultraconservative algorithms where each algorithm in the family updates its prototypes by finding a feasible solution for a set of linear constraints that depend on the instantaneous similarity-scores. We then discuss a specific online algorithm that seeks a set of prototypes which have a small norm. The resulting algorithm, which we term MIRA (for Margin Infused Relaxed Algorithm) is ultraconservative as well. We derive mistake bounds for all the algorithms and provide further analysis of MIRA using a generalized notion of the margin for multiclass problems. We discuss the form the algorithms take in the binary case and show that all the algorithms from the first family reduce to the Perceptron algorithm while MIRA provides a new Perceptron-like algorithm with a margin-dependent learning rate. We then return to multiclass problems and describe an analogous multiplicative family of algorithms with corresponding mistake bounds. We end the formal part by deriving and analyzing a multiclass version of Li and Long's ROMMA algorithm. We conclude with a discussion of experimental results that demonstrate the merits of our algorithms.

1. Introduction

In this paper we present a general approach for deriving algorithms for multiclass prediction problems. In multiclass problems the goal is to assign one of k labels to each input instance. Many machine learning problems can be phrased as a multiclass categorization problem. Examples to such problems include optical character recognition (OCR), text classification, and medical analysis. There are numerous specialized solutions for multiclass problems for specific models such as decision trees (Breiman et al., 1984, Quinlan, 1993) and neural networks. Another general approach is based on reducing a multiclass problem to multiple binary problems using output coding (Dietterich and Bakiri, 1995, Allwein et al., 2000). An example of a reduction that falls into the above framework is the "one-against-rest" approach. In one-against-rest a set of binary classifiers is trained, one classifier for each class. The rth classifier is trained to discriminate between the rth

class and the rest of the classes. New instances are classified by setting the predicted label to be the index of the classifier attaining the highest confidence in its prediction. In this paper we present a unified approach that operates directly on the multiclass problem by imposing constraints on the updates for the various classes. Thus, our approach is inherently different from methods based on output coding.

Our framework for analyzing the algorithms is the mistake bound model (Littlestone, 1988). The algorithms we study work in rounds. On each round the proposed algorithms get a new instance and output a prediction for the instance. They then receive the correct label and update their predication rule in case they made a prediction error. The goal of the algorithms is to minimize the number of mistakes they made compared to the minimal number of errors that an hypothesis, built offline, can achieve.

The algorithms we consider in this paper maintain one prototype vector for each class. Given a new instance we compare each prototype to the instance by computing the similarity-score between the instance and each of the prototypes for the different classes. We then predict the class which achieves the highest similarity-score. In binary problems, this scheme reduces (under mild conditions) to a linear discriminator. After the algorithm makes a prediction it receives the correct label of the input instance and updates the set of prototypes. For a given input instance, the set of labels that attain similarity-scores higher than the score of correct label is called the *error set*. The algorithms we describe share a common feature: they all update only the prototypes from the error sets and the prototype of the correct label. We call such algorithms *ultraconservative* algorithms.

We start in Section 3 in which we provide a motivation for our framework. We do that by revisiting the well known Perceptron algorithm and give a new account of the algorithm using two prototype vectors, one for each class. We then extend the algorithm to a multiclass setting using the notion of ultraconservativeness. In Section 4 we further generalize the multiclass version of the extended Perceptron algorithm and describe a new family of ultraconservative algorithms that we obtain by replacing the Perceptron's update with a set of linear equations. We give a few illustrative examples of specific updates from this family of algorithms. Going back to the Perceptron algorithm, we show that in the binary case all the different updates reduce to the Perceptron algorithm. We finish Section 4 by deriving a mistake bound that is common to all the additive algorithms in the family. We analyze both the separable and the non-separable case.

The fact that all algorithms from Section 4 achieve the same mistake bound implies that there are some undetermined degrees of freedom. We present in Section 5 a new online algorithm that gives a unique update and is based on a relaxation of the set of linear constraints employed by the family of algorithms from Section 4. The algorithm is derived by adding an objective function that incorporates the norm of the new matrix of prototypes and minimizing it subject to a subset of the linear constraints. Following recent trend, we call the new algorithm MIRA for Margin Infused Relaxed Algorithm. We analyze MIRA and give a mistake bound related to the instantaneous margin of individual examples. This analysis leads to modification of MIRA which incorporates the margin into the update rule. We describe a simple and efficient fixed-point algorithm that efficiently computes a single update of MIRA and prove its convergence. Both MIRA and of the additive algorithms from Section 4 can be combined with kernels techniques and voting methods.

In Section 6 we derive an analogous ultraconservative family of *multiplicative* algorithms for multiclass problems. Here we describe two variants of multiplicative algorithms. The two variants differ in the way they normalize the set of prototypes. As in the additive case, we analyze both variants in the mistake bound model. Analogously to the additive family of algorithms, the

multiplicative family of algorithms reduces to Winnow (Littlestone, 1988) in the binary case. In Section 7 we combine the ultraconservative approach with Li and Long's (2002) algorithm to derive a multiclass version of it.

In Section 8 we discuss experiments with synthetic data and real datasets that compare the additive algorithms. Our experiments indicate that MIRA outperforms the other algorithms at the expense of updating its hypothesis frequently. The algorithms presented in this paper underscore a general framework for deriving ultraconservative multiclass algorithms. This framework can be used in combination with other online techniques. To conclude, we outline some of our current research directions.

Related Work A question that is common to numerous online algorithms is how to compromise the following two demands. On one hand, we want to update the classifier we learn so that it will better predict the current input instance, in particular if an error occurs when using the current classifier. On the other hand, we do not want to change the current classifier too radically, especially if it classifies well most of the previously observed instances. The good old Perceptron algorithm suggested by Rosenblatt (1958) copes with these two requirements by replacing the classifier with a linear combination of the current hyperplane and the current instance vector. Although the algorithm uses a simple update rule, it performs well on many synthetic and real-world problems. The Perceptron algorithm spurred voluminous work which clearly cannot be covered here. For an overview of numerous additive and multiplicative online algorithms see the paper by Kivinen and Warmuth (1997). We also would like to note that the a multiclass version of the Perceptron algorithm has already been provided in the widely read and cited book of Duda and Hart (1973). The multicalss version in the book is called Kesler's construction. We postpone the discussion of the relation of this construction to our family of online algorithms to Section 4. We now outline more recent research that is relevant to the work presented in this paper.

Kivinen and Warmuth (1997) presented numerous online algorithms for regression. Their algorithms are based on minimization of an objective function which is a sum of two terms. The first term is equal to the distance between the new classifier and the current classifier while the second term is the loss on the current example. The resulting update rule can be viewed as a gradient-descent method. Although multiclass classification problems are a special case of regression problems, the algorithms for regression put emphasis on smooth loss functions which might not be suitable for classification problems.

The idea of seeking a hyperplane of a small norm is a primary goal in support vector machines (Cortes and Vapnik, 1995, Vapnik, 1998). Note that for SVMs minimizing the norm of the hyperplane is equivalent to maximizing the margin of the induced linear separator. Algorithms for constructing support vector machines solve optimization problems with a quadratic objective function and linear constraints. Anlauf and Biehl (1989) and Friess, Cristianini, and Campbell (1998) suggested an alternative approach which minimizes the objective function in a gradient-decent method. The minimization can be performed by going over the sample sequentially. Algorithms with a similar approach include the Sequential Minimization Optimization (SMO) algorithm introduced by Platt (1998). SMO works on rounds, on each round it chooses two examples of the sample and minimizes the objective function by modifying variables relevant only to these two examples. While these algorithms share some similarities with the algorithmic approaches described in this paper, they were all designed for batch problems and were not analyzed in the mistake bound model.

Another approach to the problem of designing an update rule which results in a linear classifier of a small norm was suggested by Li and Long (2002). The algorithm Li and Long proposed, called ROMMA, tackles the problem by finding a hyperplane with a minimal norm under two linear constraints. The first constraint is presented so that the new classifier will classify well previous examples, while the second rule demands that the hyperplane will classify correctly the current new instance. Solving this minimization problem leads to an additive update rule with adaptive coefficients.

Grove, Littlestone, and Schuurmans (2001) introduced a general framework of quasi-additive binary algorithms, which contain the Perceptron and Winnow as special cases. Gentile (2001) proposed an extension to a subset of the quasi-additive algorithms, which uses an additive conservative update rule with decreasing learning rates.

All of the work described above is designed to solve binary classification problems. These binary classifiers can be used in a multiclass setting by reducing them to multiple binary problems using output coding such as one-against-rest. Mesterharm (1999) suggested a multiclass online algorithm which combines results from a set of sub-experts. Using this algorithm Mesterharm derives a Winnow-like algorithm and provides a corresponding mistake bound. The multiclass algorithm of Mesterharm is closely related to the multiplicative family of algorithms we present in Section 6, though our family of multiplicative algorithms is more general.

The algorithms presented in this paper are reminiscent of some of the widely used methods for constructing classifiers in multiclass problems. As mentioned above, a popular approach for solving classification problems with many classes is to learn a set of binary classifiers where each classifier is designed to separate one class from the rest of classes. If we use the Perceptron algorithm to learn the binary classifiers, we need to maintain and update one vector for each possible class. This approach shares the same form of hypothesis as the algorithms presented in this paper, which maintain one prototype per class. Nonetheless, there is one major difference between the ultraconservative algorithms we present and the one-against-rest approach. In one-against-rest we update and change each of the classifiers *independently* of the others. In fact we can construct them one after the other by re-running over the data. In contrast, ultraconservative algorithms update all the prototypes in tandem thus updating one prototype has a global effect on the other prototypes. There are situations in which there is an error due to some classes, but not all the respective prototypes should be updated. Put another way, we might perform milder changes to the set of classifiers by changing them together with the prototypes so as to achieve the same goal. As a result we get better mistake bounds and empirically better algorithms.

2. Preliminaries

The focus of this paper is online algorithms for multiclass prediction problems. We observe a sequence $(\bar{x}^1, y^1), \ldots, (\bar{x}^t, y^t), \ldots$ of instance-label pairs. Each instance $\bar{x}^t$ is in $\mathbb{R}^n$ and each label belongs to a finite set Y of size k. We assume without loss of generality that $Y = \{1, 2, \ldots, k\}$. A *multiclass classifier* is a function $H(\bar{x})$ that maps instances from $\mathbb{R}^n$ into one of the possible labels in Y. In this paper we focus on classifiers of the form $H(\bar{x}) = \arg\max_{r=1}^{k} \{\bar{M}_r \cdot \bar{x}\}$, where $\mathbf{M}$ is a $k \times n$ matrix over the reals and $\bar{M}_r \in \mathbb{R}^n$ denotes the rth row of $\mathbf{M}$. We call the inner product of $\bar{M}_r$ with the instance $\bar{x}$, the *similarity-score* for class r. Thus, the classifiers we consider in this paper set the label of an instance to be the index of the row of $\mathbf{M}$ which achieves the highest similarity-score. The margin of H on $\bar{x}$ is the difference between the similarity-score of the correct label y and the

maximum among the similarity-scores of the rest of the rows of **M**. Formally, the margin that **M** achieves on $(\bar{x},y)$ is,

$$\bar{M}_y \cdot \bar{x} - \max_{r \neq y} \{\bar{M}_r \cdot \bar{x}\} \ .$$

The l_p norm of a vector $\bar{u} = (u_1, \ldots, u_l)$ in $\mathbb{R}^l$ is

$$\|\bar{u}\|_p = \left(\sum_{i=1}^{l} |u_i|^p \right)^{\frac{1}{p}} \ .$$

norm of the vector we get by concatenating the rows of **A**, that is,

$$\|\mathbf{A}\|_p = \|(\bar{A}_1, \ldots, \bar{A}_k)\|_p \ ,$$

where for $p = 2$ the norm is known as the Frobenius norm. Similarly, we define the vector-scalar-product of two matrices **A** and **B** to be,

$$\mathbf{A} \cdot \mathbf{B} = \sum_r \bar{A}_r \cdot \bar{B}_r \ .$$

Finally, $\delta_{i,j}$ denotes Kronecker's delta function, that is, $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ otherwise.

The framework that we use in this paper is the mistake bound model for online learning. The algorithms we consider work in rounds. On round t an online learning algorithm gets an instance $\bar{x}^t$. Given $\bar{x}^t$, the learning algorithm outputs a prediction, $\hat{y}^t = \arg\max_r \{\bar{M}_r \cdot \bar{x}^t\}$. It then receives the correct label y^t and updates its classification rule by modifying the matrix **M**. We say that the algorithm made a (multiclass) prediction error if $\hat{y}^t \neq y^t$. Our goal is to make as few prediction errors as possible. When the algorithm makes a prediction error there might be more than one row of **M** achieving a score higher than the score of the row corresponding to the correct label. We define the *error-set* for $(\bar{x},y)$ using a matrix **M** to be the index of all the rows in **M** which achieve such high scores. Formally, the error-set for a matrix **M** on an instance-label pair $(\bar{x},y)$ is,

$$E = \{r \neq y : \bar{M}_r \cdot \bar{x} \geq \bar{M}_y \cdot \bar{x}\} \ .$$

Many online algorithms update their prediction rule only on rounds on which they made a prediction error. Such algorithms are called *conservative*. We now give a definition that extends the notion of conservativeness to multiclass settings.

Definition 1 (Ultraconservative) *An online multiclass algorithm of the form* $H(\bar{x}) = \arg\max_r \{\bar{M}_r \cdot \bar{x}\}$ *is ultraconservative if it modifies* **M** *only when the error-set* E *for* $(\bar{x},y)$ *is not empty and the indices of the rows that are modified are from* $E \cup \{y\}$.

Note that our definition implies that an ultraconservative algorithm is also conservative. For binary problems the two definitions coincide.

3. From Binary to Multiclass

The Perceptron algorithm of Rosenblatt (1958) is a well known online algorithm for binary classification problems. The algorithm maintains a weight vector $\bar{w} \in \mathbb{R}^n$ that is used for prediction. To motivate our multiclass algorithms let us now describe the Perceptron algorithm using the notation

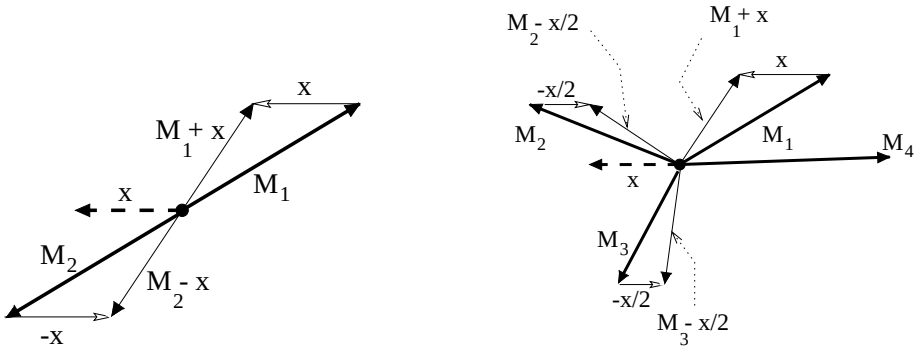

Figure 1: A geometrical illustration of the update for a binary problem (left) and a four-class problem (right) using the extended Perceptron algorithm.

employed in this paper. In our setting the label of each instance belongs to the set $\{1,2\}$. Given an input instance $\bar{x}$ the Perceptron algorithm predicts that its label is $\hat{y} = 1$ iff $\bar{w} \cdot \bar{x} \geq 0$ and otherwise it predicts $\hat{y} = 2$. The algorithm modifies $\bar{w}$ only on rounds with prediction errors and is thus conservative. On such rounds $\bar{w}$ is changed to $\bar{w} + \bar{x}$ if the correct label is $y = 1$ and to $\bar{w} - \bar{x}$ if $y = 2$.

To implement the Perceptron algorithm using a prototype matrix $\mathbf{M}$ with one row (prototype) per class, we set the first row $\bar{M}_1$ to $\bar{w}$ and the second row $\bar{M}_2$ to $-\bar{w}$. We now modify $\mathbf{M}$ every time the algorithm mis-classifies $\bar{x}$ as follows. If the correct label is 1 we replace $\bar{M}_1$ with $\bar{M}_1 + \bar{x}$ and $\bar{M}_2$ with $\bar{M}_2 - \bar{x}$. Similarly, we replace $\bar{M}_1$ with $\bar{M}_1 - \bar{x}$ and $\bar{M}_2$ with $\bar{M}_2 + \bar{x}$ when the correct label is 2 and $\bar{x}$ is misclassified. Thus, the row $\bar{M}_y$ is moved toward the misclassified instance $\bar{x}$ while the other row is moved away from $\bar{x}$. Note that this update implies that the total change to the two prototypes is zero. An illustration of this geometrical interpretation is given on the left-hand side of Figure 1. It is straightforward to verify that the algorithm is equivalent to the Perceptron algorithm.

We can now use this interpretation and generalize the Perceptron algorithm to multiclass problems as follows. For k classes we maintain a matrix $\mathbf{M}$ of k rows, one row per class. For each input instance $\bar{x}$, the multiclass generalization of the Perceptron calculates the similarity-score between the instance and each of the k prototypes. The predicted label, $\hat{y}$, is the index of the row (prototype) of $\mathbf{M}$ which achieves the highest score, that is, $\hat{y} = \arg\max_r \{\bar{M}_r \cdot \bar{x}\}$. If $\hat{y} \neq y$ the algorithm moves $\bar{M}_y$ toward $\bar{x}$ by replacing $\bar{M}_y$ with $\bar{M}_y + \bar{x}$. In addition, the algorithm moves each row $\bar{M}_r$ $(r \neq y)$ for which $\bar{M}_r \cdot \bar{x} \geq \bar{M}_y \cdot \bar{x}$ away from $\bar{x}$. The indices of these rows constitute the error set E. The algorithms presented in this paper, and in particular the multiclass version of the Perceptron algorithm, modify $\mathbf{M}$ such that the following property holds: The total change in units of $\bar{x}$ in the rows of $\mathbf{M}$ that are moved away from $\bar{x}$ is equal to the change of $\bar{M}_y$, (in units of $\bar{x}$). Specifically, for the multiclass Perceptron we replace $\bar{M}_y$ with $\bar{M}_y + \bar{x}$ and for each r in E we replace $\bar{M}_r$ with $\bar{M}_r - \bar{x}/|E|$. A geometric illustration of this update is given in the right-hand side of Figure 1. There are four classes in the example appearing in the figure. The correct label of $\bar{x}$ is $y = 1$ and since $\bar{M}_1$ is not the most similar vector to $\bar{x}$, it is moved toward $\bar{x}$. The rows $\bar{M}_2$ and $\bar{M}_3$ are also modified by subtracting $\bar{x}/2$ from each one. The last row $\bar{M}_4$ is not in the error-set since $\bar{M}_1 \cdot \bar{x} > \bar{M}_4 \cdot \bar{x}$ and therefore it is not modified. We defer the analysis of the algorithm to the next section in which we describe and analyze a family of online multiclass algorithms that also includes this algorithm.

4. A Family of Additive Multiclass Algorithms

We describe a family of ultraconservative algorithms by using the algorithm of the previous section as our starting point. The algorithm is ultraconservative and thus updates $\mathbf{M}$ only on rounds with predictions errors. The row $\bar{M}_y$ is changed to $\bar{M}_y + \bar{x}$ while for each $r \in E$ we modify $\bar{M}_r$ to $\bar{M}_r - \bar{x}/|E|$. Let us introduce a vector of weights $\bar{\tau} = (\tau_1, \ldots, \tau_k)$ and rewrite the update of the rth row as $\bar{M}_r + \tau_r \bar{x}$. Thus, for $r = y$ we have $\tau_r = 1$, for $r \in E$ we set $\tau_r = -1/|E|$, and for $r \notin E \cup \{y\}$, τ_r is zero. The weights $\bar{\tau}$ were chosen such that the total change of the rows of $\mathbf{M}$ whose indices are from E are equal to the change in $\bar{M}_y$, that is, $1 = \tau_y = -\sum_{r \in E} \tau_r$. If we do not impose the condition that for $r \in E$ all the τ_r's attain the same value, then the constraints on $\bar{\tau}$ become $\sum_{r \in E \cup \{y\}} \tau_r = 0$. This constraint enables us to move the prototypes from the error-set E away from $\bar{x}$ in different proportions as long as the total change is sum to one. The result is a whole family of multiclass algorithms. A pseudo-code of the family of algorithms is provided in Figure 2. Note that the constraints on $\bar{\tau}$ are redundant and we could have used less constraints. We make use of this more elaborate set of constraints in the next section.

Before analyzing the family of algorithms we have just introduced, we give a few examples of specific schemes to set $\bar{\tau}$. We have already described one update above which sets $\bar{\tau}$ to,

$$\tau_r = \begin{cases} -\frac{1}{|E|} & r \in E \\ 1 & r = y \\ 0 & \text{otherwise} \end{cases} .$$

Since all the τ's for rows in the error-set are equal, we call this the *uniform* multiclass update. We can also be further conservative and modify in addition to $\bar{M}_y$ only one other row in $\mathbf{M}$. A reasonable choice is to modify the row that achieves the highest similarity-score. That is, we set $\bar{\tau}$ to,

$$\tau_r = \begin{cases} -1 & r = \arg\max_s \{\bar{M}_s \cdot \bar{x}\} \\ 1 & r = y \\ 0 & \text{otherwise} \end{cases} .$$

We call this form of updating $\bar{\tau}$ the *max-score* multiclass update. The two examples above set τ_r for $r \in E$ to a fixed value, ignoring the actual values of similarity-scores each row achieves. We can also set $\bar{\tau}$ in promotion to the excess in the similarity-score of each row in the error set (with respect to $\bar{M}_y$). For instance, we can set $\bar{\tau}$ to be,

$$\tau_r = \begin{cases} -\frac{[\bar{M}_r \cdot \bar{x} - \bar{M}_y \cdot \bar{x}]_+}{\sum_{r=1}^k [\bar{M}_r \cdot \bar{x} - \bar{M}_y \bar{x}]_+} & r \neq y \\ 1 & r = y \end{cases} ,$$

where $[x]_+$ is equal to x if $x \geq 0$ and zero otherwise. Note that the above update implies that $\tau_r = 0$ for $r \notin E \cup \{y\}$.

We describe experiments comparing the above updates in Section 8. We proceed to analyze the family of algorithms.

4.1 Analysis

Before giving the analysis of the algorithms of Figure 2 we prove the following auxiliary lemma.

Lemma 2 *For any set $\{\tau_1, \ldots, \tau_k\}$ such that, $\sum_{r=1}^k \tau_r = 0$ and $\tau_r \leq \delta_{r,y}$ for $r = 1, \ldots, k$, then $\sum_r \tau_r^2 \leq 2\tau_y \leq 2$.*

Initialize: Set $\mathbf{M} = 0$ $(\mathbf{M} \in \mathbb{R}^{k \times n})$.
Loop: For $t = 1, 2, \ldots, T$

- Get a new instance $\vec{x}^t \in \mathbb{R}^n$.
- Predict $\hat{y}^t = \arg\max_{r=1}^{k} \{\bar{M}_r \cdot \vec{x}^t\}$.
- Get a new label y^t.
- Set $E = \{r \neq y^t \ : \bar{M}_r \cdot \vec{x}^t \geq \bar{M}_{y^t} \cdot \vec{x}^t\}$.
- If $E \neq \emptyset$ update $\mathbf{M}$ by choosing any $\tau_1^t, \ldots, \tau_k^t$ that satisfy:
  1. $\tau_r^t \leq 0$ for $r \neq y^t$ and $\tau_{y^t}^t \leq 1$.
  2. $\sum_{r=1}^k \tau_r^t = 0$.
  3. $\tau_r^t = 0$ for $r \notin E \cup \{y^t\}$.
  4. $\tau_{y^t}^t = 1$.
- For $r = 1, 2, \ldots, k$ update: $\bar{M}_r \leftarrow \bar{M}_r + \tau_r^t \vec{x}^t$.

Output : $H(\bar{x}) = \arg\max_r \{\bar{M}_r \cdot \bar{x}\}$.

Figure 2: A family of additive multiclass algorithms.

Proof Since for $r \neq y$ the value of τ_r cannot be positive we have,

$$\|\bar{\tau}\|_1 = \sum_{r=1}^{k} |\tau_r| = \tau_y + \sum_{r \neq y}^{k} (-\tau_r)$$

Using the equality $\sum_{r=1}^k \tau_r = 0$ we get,

$$\|\bar{\tau}\|_1 = 2\tau_y \ .$$

Applying Hölder's inequality we get,

$$\sum_{r=1}^{k} \tau_r^2 = \sum_{r=1}^{k} (\tau_r \tau_r) \leq \|\bar{\tau}\|_1 \, \|\bar{\tau}\|_\infty = 2\tau_y \tau_y \leq 2\tau_y \leq 2 \ ,$$

where for the last two inequalities we used the fact that $0 \leq \tau_y \leq 1$. ■

We now give the main theorem of this section.

Theorem 3 *Let $(\vec{x}^1, y^1), \ldots, (\vec{x}^T, y^T)$ be an input sequence for any multiclass algorithm from the family described in Figure 2 where $\vec{x}^t \in \mathbb{R}^n$ and $y^t \in \{1, 2, \ldots, k\}$. Denote by $R = \max_t \|\vec{x}^t\|$. Assume that there is a matrix $\mathbf{M}^*$ of a unit vector-norm, $\|\mathbf{M}^*\| = 1$, that classifies the entire sequence correctly with margin*

$$\gamma = \min_t \{\bar{M}^*_{y^t} \cdot \vec{x}^t - \max_{r \neq y^t} \bar{M}^*_r \cdot \vec{x}^t\} > 0 \ .$$

Then, the number of mistakes that the algorithm makes is at most

$$2\frac{R^2}{\gamma^2} \ .$$

Proof Assume that an error occurred when classifying the tth example $(\vec{x}^t, y^t)$ using the matrix $\mathbf{M}$. Denote by $\mathbf{M}'$ the updated matrix after round t. That is, for $r = 1, 2, \ldots, k$ we have $\bar{M}'_r = \bar{M}_r + \tau^t_r \vec{x}^t$. To prove the theorem we bound $\|\mathbf{M}\|_2^2$ from above and below. First, we derive a lower bound on $\|\mathbf{M}\|^2$ by bounding the term,

$$\begin{aligned}\sum_{r=1}^{k} \bar{M}^*_r \cdot \bar{M}'_r &= \sum_{r=1}^{k} \bar{M}^*_r \cdot (\bar{M}_r + \tau^t_r \vec{x}^t) \\ &= \sum_{r=1}^{k} \bar{M}^*_r \cdot \bar{M}_r + \sum_r \tau^t_r \left(\bar{M}^*_r \cdot \vec{x}^t\right) . \end{aligned} \tag{1}$$

We further develop the second term of Equation (1) using the second constraint of the algorithm $\left(\sum_{r=1}^{k} \tau^t_r = 0\right)$. Substituting $\tau_{y^t} = -\sum_{r \neq y^t} \tau^t_r$ we get,

$$\begin{aligned}\sum_r \tau^t_r \left(\bar{M}^*_r \cdot \vec{x}^t\right) &= \sum_{r \neq y^t} \tau^t_r \left(\bar{M}^*_r \cdot \vec{x}^t\right) + \tau_{y^t} \left(\bar{M}^*_{y^t} \cdot \vec{x}^t\right) \\ &= \sum_{r \neq y^t} \tau^t_r \left(\bar{M}^*_r \cdot \vec{x}^t\right) - \sum_{r \neq y^t} \tau^t_r \left(\bar{M}^*_{y^t} \cdot \vec{x}^t\right) \\ &= \sum_{r \neq y^t} \left(-\tau^t_r\right) \left(\bar{M}^*_{y^t} - \bar{M}^*_r\right) \cdot \vec{x}^t . \end{aligned} \tag{2}$$

Using the assumption that $\mathbf{M}^*$ classifies each instance with a margin of at least γ and that $\tau_y = 1$ (fourth constraint) we obtain,

$$\sum_r \tau^t_r \left(\bar{M}^*_r \cdot \vec{x}^t\right) \geq \sum_{r \neq y^t} \left(-\tau^t_r\right) \gamma = \tau^t_{y^t} \gamma = \gamma . \tag{3}$$

Combining Equation (1) and Equation (3) we get,

$$\sum_r \bar{M}^*_r \cdot \bar{M}'_r \geq \sum_r \bar{M}^*_r \cdot \bar{M}_r + \gamma .$$

Thus, if the algorithm made m mistakes in T rounds then the matrix $\mathbf{M}$ satisfies,

$$\sum_r \bar{M}^*_r \cdot \bar{M}_r \geq m\gamma . \tag{4}$$

Using the vector-norm definition and applying the Cauchy-Schwartz inequality we get,

$$\begin{aligned}\|\mathbf{M}^*\|^2 \|\mathbf{M}\|^2 &= \left(\sum_{r=1}^{k} \|\bar{M}^*_r\|^2\right) \left(\sum_{r=1}^{k} \|\bar{M}_r\|^2\right) \\ &\geq \left(\bar{M}^*_1 \cdot \bar{M}_1 + \ldots + \bar{M}^*_k \cdot \bar{M}_k\right)^2 \\ &= \left(\sum_{r=1}^{k} \bar{M}^*_r \cdot \bar{M}_r\right)^2 . \end{aligned} \tag{5}$$

Plugging Equation (4) into Equation (5) and using the assumption that $\mathbf{M}^*$ is of a unit vector-norm we get the following lower bound,

$$\|\mathbf{M}\|^2 \geq m^2 \gamma^2 . \tag{6}$$

Next, we bound the vector-norm of $\mathbf{M}$ from above. As before, assume that an error occurred when classifying the example $(\vec{x}^t, y^t)$ using the matrix $\mathbf{M}$ and denote by $\mathbf{M}'$ the matrix after the update. Then,

$$\begin{aligned}\|\mathbf{M}'\|^2 = \sum_r \|\bar{M}'_r\|^2 &= \sum_r \|\bar{M}_r + \tau_r^t \vec{x}^t\|^2 \\ &= \sum_r \|\bar{M}_r\|^2 + 2\sum_r \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) + \sum_r \|\tau_r^t \vec{x}^t\|^2 \\ &= \|\mathbf{M}\|^2 + 2\sum_r \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) + \|\vec{x}^t\|^2 \sum_r (\tau_r^t)^2 \ . \end{aligned} \tag{7}$$

We further develop the second term using the second constraint of the algorithm and analogously to Equation (2) we get,

$$\sum_r \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) = \sum_{r \neq y^t} (-\tau_r^t) \left(\bar{M}_{y^t} - \bar{M}_r\right) \cdot \vec{x}^t \ .$$

Since $\vec{x}^t$ was misclassified we need to consider the following two cases. The first case is when the label r was not the source of the error, that is $(\bar{M}_{y^t} - \bar{M}_r) \cdot \vec{x}^t > 0$. Then, using the third constraint $(r \notin E \cup \{y^t\} \Rightarrow \tau_r^t = 0)$ we get that $\tau_r^t = 0$ and thus $(-\tau_r^t)\left(\bar{M}_{y^t} - \bar{M}_r\right) \cdot \vec{x}^t = 0$. The second case is when one of the sources of error was the label r. In that case $(\bar{M}_{y^t} - \bar{M}_r) \cdot \vec{x}^t \leq 0$. Using the first constraint of the algorithm we know that $\tau_r^t \leq 0$ and thus $(-\tau_r^t)\left(\bar{M}_{y^t} - \bar{M}_r\right) \cdot \vec{x}^t \leq 0$. Finally, summing over all r we get,

$$\sum_r \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) \leq 0 \ . \tag{8}$$

Plugging Equation (8) into Equation (7) we get,

$$\|\mathbf{M}'\|^2 \leq \|\mathbf{M}\|^2 + \|\vec{x}^t\|^2 \sum_r (\tau_r^t)^2 \ .$$

Using the bound $\|\vec{x}^t\| \leq R$ and Lemma 2 we obtain,

$$\|\mathbf{M}'\|^2 \leq \|\mathbf{M}\|^2 + 2\|R\|^2 \ . \tag{9}$$

Thus, if the algorithm made m mistakes in T rounds, the matrix $\mathbf{M}$ satisfies,

$$\|\mathbf{M}\|^2 \leq 2m\|R\|^2 \ . \tag{10}$$

Combining Equation (6) and Equation (10), we have that,

$$m^2\gamma^2 \leq \|\mathbf{M}\|^2 \leq 2m\|R\|^2 \ ,$$

and therefore,

$$m \leq 2\frac{R^2}{\gamma^2} \ . \tag{11}$$

■

We would like to note that the bound of the above theorem reduces to the Perceptron's mistake bound in the binary case ($k = 2$). To conclude this section we analyze the non-separable case by generalizing Theorem 2 of Freund and Schapire (1999) to a multiclass setting. The proof technique follows the proof outline of Freund and Schapire and is given in Appendix A.

Initialize: Set $\mathbf{M} \neq 0$ $\mathbf{M} \in \mathbb{R}^{k \times n}$.
Loop: For $t = 1, 2, \ldots, T$

- Get a new instance $\vec{x}^t$.
- Predict $\hat{y}^t = \arg\max_r \{\bar{M}_r \cdot \vec{x}^t\}$.
- Get a new label y^t.
- Find $\vec{\tau}^t$ that solves the following optimization problem:

$$\begin{array}{ll} \min_{\vec{\tau}} & \frac{1}{2}\sum_r \|\bar{M}_r + \tau_r \vec{x}^t\|_2^2 \\ \text{subject to :} & (1)\ \tau_r \leq \delta_{r,y^t} \text{ for } r = 1, \ldots, k \\ & (2)\ \sum_{r=1}^k \tau_r = 0 \end{array}$$

- Update : $\bar{M}_r \leftarrow \bar{M}_r + \tau_r^t \vec{x}^t$ for $r = 1, 2, \ldots, k$.

Output : $H(\vec{x}) = \arg\max_r \{\bar{M}_r \cdot \vec{x}\}$.

Figure 3: The Margin Infused Relaxed Algorithm (MIRA).

Theorem 4 *Let $(\vec{x}^1, y^1), \ldots, (\vec{x}^T, y^T)$ be an input sequence for any multiclass algorithm from the family described in Figure 2, where $\vec{x}^t \in \mathbb{R}^n$ and $y^t \in \{1, 2, \ldots, k\}$. Denote by $R = \max_t \|\vec{x}^t\|$. Let $\mathbf{M}^*$ be a prototype matrix of a unit vector-norm, $\|\mathbf{M}^*\| = 1$, and fix some $\gamma > 0$. Define,*

$$d^t = \max\left\{0,\ \gamma - \left[\bar{M}^*_{y^t} \cdot \vec{x}^t - \max_{r \neq y^t} \bar{M}^*_r \cdot \vec{x}^t\right]\right\} ,$$

and denote by $D^2 = \sum_{t=1}^T (d^t)^2$. Then the number of mistakes the algorithm makes is at most

$$2\frac{(R+D)^2}{\gamma^2} .$$

4.2 The Relation to Kesler's Construction

Before turning to a more complex multiclass version, we would like to discuss the relation of the family of updates described in this section to Kesler's construction (Duda and Hart, 1973). Kesler's construction is attributed to Carl Kesler and was described by Nilsson (1965). The construction reduces a multiclass classification problem to a binary problem by expanding each instance in $\mathbb{R}^n$ into an instance $\mathbb{R}^{n(k-1)}$. By unravelling Kesler's expansion the resulting update in the original space amounts to a succession of our *max* update. Specifically, the update due to Kesler is ultraconservative as it modifies only the prototypes whose indices constitute the error set. Given an example $(\vec{x}^t, y^t)$ Kesler's update rule cycles through the labels $y \neq y^t$ and if $\bar{M}^t_y \cdot \vec{x}^t > \bar{M}^t_{y^t} \cdot \vec{x}^t$ it applies the max-update to the prototypes indexed y and y^t. Therefore, the family of online algorithms presented thus far is a generalization of Kesler's construction in terms of the form of the specific update.

5. A Norm-Optimized Multiclass Algorithm

In the previous section we have described a family of algorithms where each algorithm of the family achieves the same mistake bound given by Theorem 3 and Theorem 4. This variety of equivalent

algorithms suggests that there are some degrees of freedom that we might be able to exploit. In this section we describe an online algorithm that chooses a feasible vector $\overline{\tau}^t$ such that the vector-norm of the matrix $\mathbf{M}$ will be as small as possible.

To derive the new algorithm we omit the forth constraint ($\tau_y = 1$) and thus allow more flexibility in choosing $\overline{\tau}^t$, or smaller changes in the prototype matrix. Previous bounds provide motivation for the algorithms in this section. We choose a vector $\overline{\tau}^t$ which minimizes the vector-norm of the new matrix $\mathbf{M}$ subject to the first two constraints only. As we show in the sequel, the solution of the optimization problem automatically satisfies the third constraint. The algorithm attempts to update the matrix $\mathbf{M}$ on *each* round regardless of whether there was a prediction error or not. We show below that the algorithm is ultraconservative and thus $\overline{\tau}^t$ is the zero vector if $\overline{x}^t$ is correctly classified (and no update takes place). Following the trend paved by Li and Long (2002) and Gentile (2001), we term our algorithm MIRA for Margin Infused Relaxed Algorithm. The algorithm is described in Figure 3.

Before investigating the properties of the algorithm, we rewrite the optimization problem that MIRA solves on each round in a more convenient form. Omitting the example index t the objective function becomes,

$$\frac{1}{2}\sum_r \|\overline{M}_r + \tau_r \overline{x}\|^2 \quad = \quad \frac{1}{2}\sum_r \|\overline{M}_r\|^2 + \sum_r \tau_r \left(\overline{M}_r \cdot \overline{x}\right) + \frac{1}{2}\sum_r \tau_r^2 \|\overline{x}\|^2 \ .$$

Omitting $\frac{1}{2}\sum_r \|\overline{M}_r\|^2$ which is constant, the quadratic optimization problem becomes,

$$\begin{aligned} &\min_{\tau} \ Q\left(\overline{\tau}\right) = \frac{1}{2} A \sum_{r=1}^{k} \tau_r^2 + \sum_{r=1}^{k} B_r \tau_r \\ &\text{subject to : } \ \forall r \quad \tau_r \leq \delta_{r,y} \quad \text{and} \quad \textstyle\sum_r \tau_r = 0 \end{aligned} \tag{12}$$

where,

$$A = \|\overline{x}\|^2 \ , \tag{13}$$

and

$$B_r = \overline{M}_r \cdot \overline{x} \ . \tag{14}$$

Since Q is a quadratic function, and thus strictly convex, and the constraints are linear, the problem has a unique solution.

We now show that MIRA automatically satisfies the third constraint of the family of algorithms from Section 4, which implies that it is ultraconservative. We first prove the following auxiliary lemma.

Lemma 5 *Let $\overline{\tau}$ be the optimal solution of the constrained optimization problem given by Equation (12) for an instance-label pair $(\overline{x}, y)$. For each $r \neq y$ such that $B_r \leq B_y$ then $\tau_r = 0$.*

Proof Assume by contradiction that there is a vector $\overline{\tau}$ which minimizes the objective function of Equation (12) and for some $s \neq y$ we have that both $B_s \leq B_y$ and $\tau_s < 0$. Note that this implies that $\tau_y > 0$. Define a new vector $\overline{\tau}'$ as follows,

$$\tau'_r = \begin{cases} 0 & r = s \\ \tau_y + \tau_s & r = y \\ \tau_r & \text{otherwise} \end{cases} \ .$$

It is easy to verify that two linear constraints of MIRA are still satisfied by $\bar{\tau}'$. Since $\bar{\tau}'$ and $\bar{\tau}$ differ only at their s and y components we get,

$$\begin{aligned} Q(\bar{\tau}') - Q(\bar{\tau}) &= \frac{1}{2}A({\tau'_s}^2 + {\tau'_y}^2) + \tau'_s B_s + \tau'_y B_y \\ &\quad - \left[\frac{1}{2}A({\tau_s}^2 + {\tau_y}^2) + \tau_s B_s + \tau_y B_y\right] . \end{aligned}$$

Expanding τ' we get,

$$\begin{aligned} Q(\bar{\tau}') - Q(\bar{\tau}) &= \frac{1}{2}A(\tau_s + \tau_y)^2 + (\tau_y + \tau_s)B_y \\ &\quad - \left[\frac{1}{2}A({\tau_s}^2 + {\tau_y}^2) + \tau_s B_s + \tau_y B_y\right] \\ &= A\tau_s\tau_y + \tau_s(B_y - B_s) . \end{aligned}$$

From the fact that $\tau_s < 0$ and the assumption $(B_s \leq B_y)$ we get that the right term is less than or equal to zero. Also, since $A\tau_y > 0$ we get that the left term is less then zero. We therefore get that $Q(\bar{\tau}') - Q(\bar{\tau}) < 0$, which contradicts the assumption that $\bar{\tau}$ is a solution of Equation (12). ■

The lemma implies that if a label r is not a source of error, then the rth prototype, $\bar{M}_r$, is not updated after $(\bar{x}, y)$ has been observed. In other words, the solution of Equation (12) satisfies that $\tau_r = 0$ for all $r \neq y$ with $(\bar{M}_r \cdot \bar{x} \leq \bar{M}_y \cdot \bar{x})$.

Corollary 6 *MIRA is ultraconservative.*

Proof Let $(\bar{x}, y)$ be a new example fed to the algorithm. And let $\bar{\tau}$ be the coefficients found by the algorithm. From Lemma 5 we get that for each label r whose score $(\bar{M}_r \cdot \bar{x})$ is not larger than the score of the correct label $(\bar{M}_y \cdot \bar{x})$ its corresponding value τ_r is set to zero. This implies that only the indices which belong to the set $E \cup \{y\} = \{r \neq y : \bar{M}_r \cdot \bar{x} \geq \bar{M}_y \cdot \bar{x}\} \cup \{y\}$ may be updated. Furthermore, if the algorithm predicts correctly that the label is y, we get that $E = \emptyset$ and $\tau_r = 0$ for all $r \neq y$. In this case τ_y is set to zero due to the constraint $\sum_r \tau_r = \tau_y + \sum_{r \neq y} \tau_r = 0$. Hence, $\bar{\tau} = 0$ and the algorithm does not modify $\mathbf{M}$ on $(\bar{x}, y)$. Thus, the conditions required for ultraconservativeness are satisfied. ■

In Section 5.3 we give a detailed analysis of MIRA that incorporates the margin achieved on each example, and can be used to derive a mistake bound. Let us first show that the cumulative l_1-norm of the coefficients $\bar{\tau}^t$ is bounded.

Theorem 7 *Let $(\bar{x}^1, y^1), \ldots, (\bar{x}^T, y^T)$ be an input sequence to MIRA where $\bar{x}^t \in \mathbb{R}^n$ and $y^t \in \{1, 2, \ldots, k\}$. Let $R = \max_t \|\bar{x}^t\|$ and assume that there is a prototype matrix $\mathbf{M}^*$ of a unit vector-norm, $\|\mathbf{M}^*\| = 1$, which classifies the entire sequence correctly with margin $\gamma = \min_t \{\bar{M}^*_{y^t} \cdot \bar{x}^t - \max_{r \neq y^t} \bar{M}^*_r \cdot \bar{x}^t\} > 0$. Let $\bar{\tau}^t$ be the coefficients that MIRA finds for $(\bar{x}^t, y^t)$. Then, the following bound holds,*

$$\sum_{t=1}^{T} \|\bar{\tau}^t\|_1 \leq 4\frac{R^2}{\gamma^2} .$$

The proof employs the technique used in the proof of Theorem 3. The proof is given for completeness in Appendix A.

5.1 Characteristics of the Solution

Let us now further examine the characteristics of the solution obtained by MIRA. In a recent paper (Crammer and Singer, 2000) we investigated a related setting that uses error correcting output codes for multiclass problems. Using these results, it is simple to show that the optimal $\bar{\tau}$ in Equation (12) is given by

$$\tau_r = \min\{\theta^* - \frac{B_r}{A}, \delta_{y,r}\} \ , \tag{15}$$

where $A = \|\bar{x}\|^2$ and $B_r = \bar{M}_r \cdot \bar{x}$ is the similarity-score of $(\bar{x}, y)$ for label r, as defined by Equation (13) and Equation (14), respectively. The optimal value θ^* is uniquely defined by the equality constraint $\sum_r \tau_r = 0$ of Equation (12) and satisfies,

$$\sum_{r=1}^{k} \min\{\theta^* - \frac{B_r}{A}, \delta_{y,r}\} = 0 \ .$$

The value θ^* can be found by a binary search (Crammer and Singer, 2000) or iteratively by solving a fixed point equation (Crammer and Singer, 2001).

We now can view MIRA in the following alternative light. Assume that the instance $(\bar{x}, y)$ was misclassified by MIRA and set $E = \{r \neq y \ : \bar{M}_r \cdot \bar{x} \geq \bar{M}_y \cdot \bar{x}\} \neq \emptyset$. The similarity-score for label r of the updated matrix on the current instance $\bar{x}$ is,

$$(\bar{M}_r + \tau_r \bar{x}) \cdot \bar{x} = B_r + \tau_r A \ . \tag{16}$$

Plugging Equation (15) into Equation (16) we get that the similarity-score for class r on the current instance is,

$$\min\{A\theta^*, B_r + A\delta_{y,r}\} \ .$$

Since $\tau_r \leq \delta_{y,r}$, the maximal similarity score the updated matrix can attain on $\bar{x}$ is $B_r + A\delta_{r,y}$. Thus, the similarity-score for class r after the update is either a constant that is common to all classes, $A\theta^*$, or the largest similarity-score the class r can attain, $B_r + A\delta_{r,y}$. The constant $A\theta^*$ places an upper bound on the similarity-score for all classes after the update. This bound is tight, that is at least one similarity-score value is equal to $A\theta^*$.

5.2 Using MIRA for Binary Classification Problems

In this section we discuss MIRA in the special case in which there are only two possible labels. First, note that any algorithm that belongs to the family of algorithms from Figure 2 reduces to the Perceptron algorithm in the the binary case. We now further analyze MIRA, assuming that the labels are drawn from the set $y \in \{-1, +1\}$. In this case the first row of $\mathbf{M}$ corresponds to the label $y = +1$ and the second row to the label $y = -1$. We now derive the equations for the case $y = +1$. The case $y = -1$ is derived similarly by replacing the indices 1 and 2 in all the equations. The constraints of MIRA reduce to $\tau_1 \leq 1$, $\tau_2 \leq 0$ and $\tau_1 + \tau_2 = 0$. Thus, if the algorithm is initialized with a matrix $\mathbf{M}$ such that $\bar{M}_1 + \bar{M}_2 = 0$, this property is conserved along its execution. Therefore, we can replace the matrix $\mathbf{M}$ with a single vector $\bar{w}$ such that $\bar{M}_1 = \bar{w}$ and $\bar{M}_2 = -\bar{w}$. The objective function of Equation (12) now becomes,

$$Q = \frac{1}{2}\|\bar{x}\|^2 \left(\tau_1^2 + \tau_2^2\right) + y(\bar{w} \cdot \bar{x})\tau_1 + y(-\bar{w} \cdot \bar{x})\tau_2 \ .$$

Initialize: Set $\bar{w} \neq 0$.
Loop: For $t = 1, 2, \ldots, T$

- Get a new instance $\bar{x}^t$.
- Predict $\hat{y}^t = \text{sign}(\bar{w} \cdot \bar{x}^t)$.
- Get a new label $y^t \in \{-1, +1\}$.
- Define $\tau^t = G\left(-\frac{y^t(\bar{w} \cdot \bar{x}^t)}{\|\bar{x}^t\|^2}\right)$ where:

$$G(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & 1 < x \end{cases}$$

- Update: $\bar{w} \leftarrow \bar{w} + \tau^t y^t \bar{x}^t$

Output : $H(\bar{x}) = \text{sign}(\bar{w} \cdot \bar{x})$.

Figure 4: Binary MIRA.

We now omit the label index and identify τ with τ_1 and $-\tau$ with τ_2 to get the following optimization problem,

$$\begin{aligned} \min_{\tau} \quad & Q = \|\bar{x}\|^2 \tau^2 + 2y(\bar{w} \cdot \bar{x})\tau \\ \text{subject to :} \quad & 0 \leq \tau \leq 1 \ . \end{aligned} \tag{17}$$

It is easy to verify that the solution of this problem is given by,

$$\tau = G\left(-\frac{y(\bar{w} \cdot \bar{x})}{\|\bar{x}\|^2}\right) \ , \tag{18}$$

where

$$G(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & 1 < x \end{cases} \ .$$

Clearly, the binary version of MIRA is conservative since if $\bar{x}$ is classified correctly $\left(\frac{y(\bar{w} \cdot \bar{x})}{\|\bar{x}\|^2} > 0\right)$ then $\bar{w}$ is not modified. Furthermore, the coefficient τ is equal to the absolute value of the normalized margin $y(\bar{w} \cdot \bar{x})/\|\bar{x}\|^2$, as long as this normalized margin is smaller than one. The bound on the norm ensures that a new example does not change the prediction vector $\bar{w}$ too radically, even if the margin is a large negative number. The algorithm is described in Figure 4. Note that the algorithm is very similar to the Perceptron algorithm. The only difference between binary MIRA and the Perceptron is the function used for determining the value of τ. For the Perceptron we use the function

$$S(x) = \begin{cases} 0 & x \leq 0 \\ 1 & 0 < x \end{cases} \ .$$

instead of $G(x)$. One interesting question that comes to mind is whether we can use other functions of the normalized margin to derive other online algorithms with corresponding mistake bounds. We leave this for future research.

5.3 Margin Analysis of MIRA

In this section we further analyze MIRA by relating its mistake bound to the instantaneous margin of the individual examples. Note that since MIRA was derived from the family of algorithms in Figure 2 by dropping the fourth constraint. Therefore, Theorem 3 and 4 do no hold and we thus need to derive an alternative analysis. The margin analysis we present in this section sheds some more light on the source of difficulty in achieving a mistake bound for MIRA. Our analysis here also leads to an alternative version of MIRA that incorporates the margin into the quadratic optimization problem that we need to solve on each round. Our starting point is Theorem 7. We first give a lower bound on τ_y on each round. If MIRA made a mistake on $(\bar{x},y)$, then we know that $\max_{r\neq y} B_r - B_y > 0$, where $B_r = \bar{M}_r \cdot \bar{x}$ (see Equation (14)). Therefore, we can bound the minimal value of τ_y by a function of the (negative) margin, $B_y - \max_{r\neq y} B_r$.

Lemma 8 *Let $\bar{\tau}$ be the optimal solution of the constrained optimization problem given by Equation (12) for an instance-label pair $(\bar{x},y)$ with $A \leq R^2$. Assume that the margin $B_y - \max_{r\neq y} B_r$ is bounded from above by $-\beta$, where $0 < \beta \leq 2R^2$. Then τ_y is at least $\beta/(2R^2)$.*

Proof Assume by contradiction that the solution of the quadratic problem of Equation (12) satisfies $\tau_y < \beta/(2R^2)$. Note that $\tau_y > 0$ since $\max_{r\neq y} B_r - B_y \geq \beta > 0$. Let us define $\Delta = \beta/(2R^2) - \tau_y > 0$ and let $s = \arg\max_r B_r$ (ties are broken arbitrarily). Define a new vector $\bar{\tau}'$ as follows,

$$\tau'_r = \begin{cases} \tau_s - \Delta & r = s \\ \tau_y + \Delta & r = y \\ \tau_r & \text{otherwise} \end{cases} .$$

The vector $\bar{\tau}'$ satisfies the constraints of the quadratic optimization problem because $\tau'_y = \beta/(2R^2) \leq 1$. Since $\bar{\tau}'$ and $\bar{\tau}$ differ only at their s and y components we get,

$$\begin{aligned} Q(\bar{\tau}') - Q(\bar{\tau}) &= \frac{1}{2}A({\tau'_y}^2 + {\tau'_s}^2) + \tau'_y B_y + \tau'_s B_s \\ &\quad - \left[\frac{1}{2}A({\tau_y}^2 + {\tau_s}^2) + \tau_y B_y + \tau_s B_s\right] . \end{aligned}$$

Substituting $\bar{\tau}'$ we get,

$$\begin{aligned} Q(\bar{\tau}') - Q(\bar{\tau}) &= \frac{1}{2}A\left[(\tau_y+\Delta)^2 + (\tau_s-\Delta)^2\right] + B_y(\tau_y+\Delta) + B_s(\tau_s-\Delta) \\ &\quad - \left[\frac{1}{2}A({\tau_y}^2 + {\tau_s}^2) + \tau_y B_y + \tau_s B_s\right] \\ &= \Delta\left[A(\tau_y - \tau_s) + A\Delta + B_y - B_s\right] . \end{aligned}$$

Using the second constraint of MIRA ($\sum_r \tau_r = 0$) we get that $\|\bar{\tau}\|_1 = 2\tau_y$ and thus $\tau_y - \tau_s \leq 2\tau_y$. Hence,

$$Q(\bar{\tau}') - Q(\bar{\tau}) \leq \Delta\left(A(2\tau_y + \Delta) + B_y - B_s\right) .$$

Substituting $\tau_y + \Delta = \beta/(2R^2)$ and using the assumption that $\tau_y < \beta/(2R^2)$ we get,

$$Q(\bar{\tau}') - Q(\bar{\tau}) \leq \Delta\left(\frac{\beta A}{R^2} + B_y - B_s\right) .$$

Since $B_s - B_y \geq \beta$ for $(\bar{x}, y)$ we get,

$$\begin{aligned} Q(\bar{\tau}') - Q(\bar{\tau}) &\leq \Delta\left(\frac{\beta A}{R^2} - \beta\right) \\ &= \frac{\Delta\beta}{R^2}\left(A - R^2\right) . \end{aligned}$$

Finally, since $A = \|\bar{x}\|^2 \leq R^2$ and $\beta\Delta > 0$ we obtain that,

$$Q(\bar{\tau}') - Q(\bar{\tau}) \leq 0 .$$

Now, either $Q(\bar{\tau}') = Q(\bar{\tau})$, which contradicts the uniqueness of the solution, or $Q(\bar{\tau}') < Q(\bar{\tau})$ which implies that $\bar{\tau}$ is not the optimal value and again we reach a contradiction. ■

We would like to note that for the above lemma if $\beta \geq 2R^2$ then $\tau_y = 1$ regardless of the margin achieved. We are now ready to prove the main result of this section.

Theorem 9 *Let $(\bar{x}^1, y^1), \ldots, (\bar{x}^T, y^T)$ be an input sequence to MIRA where $\bar{x}^t \in \mathbb{R}^n$ and $y^t \in \{1, 2, \ldots, k\}$. Denote by $R = \max_t \|\bar{x}^t\|$ and assume that there is a prototype matrix $\mathbf{M}^*$ of a unit vector-norm, $\|\mathbf{M}^*\|_2 = 1$, which classifies the entire sequence correctly with margin $\gamma = \min_t \{\bar{M}^*_{y^t} \cdot \bar{x}^t - \max_{r \neq y^t} \bar{M}^*_r \cdot \bar{x}^t\} > 0$. Denote by n_β the number of rounds for which $B_{y^t} - \max_{r \neq y^t} B_r \leq -\beta$, for some $0 < \beta \leq 2R^2$. Then the following bound holds,*

$$n_\beta \leq 4\frac{R^4}{\beta\gamma^2} .$$

Proof The proof is a simple application of Theorem 7 and Lemma 8. Using the second constraint of MIRA ($\sum_r \tau_r = 0$) and Theorem 7 we get that,

$$\sum_{t=1}^{T} \tau^t_{y^t} \leq 2\frac{R^2}{\gamma^2} . \tag{19}$$

From Lemma 8 we know that whenever $\max_{r \neq y^t} B_r - B_{y^t} \geq \beta$ then $1 \leq \frac{2R^2}{\beta}\tau^t_{y^t}$ and therefore,

$$n_\beta \leq \sum_{t=1}^{T} \frac{2R^2}{\beta}\tau^t_{y^t} . \tag{20}$$

Combining Equation (19) and Equation (20) we obtain the required bound,

$$n_\beta \leq 2\frac{R^2}{\beta}\sum_{t=1}^{T}\tau^t_{y^t} \leq 2\frac{R^2}{\beta}2\frac{R^2}{\gamma^2} \leq 4\frac{R^4}{\beta\gamma^2} .$$

■

Note that Theorem 9 still does not provide a mistake bound for MIRA since in the limit of $\beta \to 0$ the bound diverges. Note also that for $\beta = 2R^2$ the bound reduces to the bounds of Theorem 3 and Theorem 7. The source of the difficulty in obtaining a mistake bound is rounds on which MIRA

achieves a small negative margin and thus makes small changes to $\mathbf{M}$. On such rounds τ_y can be arbitrarily small and we cannot translate the bound on $\sum_t \tau_{y^t}^t$ into a mistake bound. This implies that MIRA is not robust to small changes in the input instances. We therefore describe now a simple modification to MIRA for which we can prove a mistake bound and, as we later see, performs well empirically.

The modified MIRA aggressively updates $\mathbf{M}$ on every round for which the margin is smaller than some predefined value denoted again by β. This technique is by no means new, see for instance the paper of Li and Long (2002). The result is a mixed algorithm which is both aggressive and ultraconservative. On one hand, the algorithm updates $\mathbf{M}$ whenever a minimal margin is not achieved, including rounds on which $(\bar{x}, y)$ is classified correctly but with a small margin. On the other hand, on each update of $\mathbf{M}$ only the rows whose corresponding similarity-scores are mistakenly too high are updated. We now describe how to modify MIRA along these lines.

To achieve a minimal margin of at least $\beta \leq 2R^2$ we modify the optimization problem given by Equation (12). A minimal margin of β is achieved if for all r we require $\bar{M}_y \cdot \bar{x} - \bar{M}_r \cdot \bar{x} \geq \beta$ or, alternatively, $(\bar{M}_y \cdot \bar{x} - \beta) - (\bar{M}_r \cdot \bar{x}) \geq 0$. Thus, if we replace B_y with $B_y - \beta$, $\mathbf{M}$ will be updated whenever the margin is smaller than β. We thus let MIRA solve for each example $(\bar{x}, y)$ the following constrained optimization problem,

$$\min_{\tau} \; Q(\bar{\tau}) = \frac{1}{2}\tilde{A}\sum_{r=1}^{k}\tau_r^2 + \sum_{r=1}^{k}\tilde{B}_r\tau_r$$
$$\text{subject to : } \; \forall r \quad \tau_r \leq \delta_{r,y} \quad \text{and} \quad \sum_r \tau_r = 0$$

$$\text{where : } \; \tilde{A} = A = \|\bar{x}\|^2 \; ; \; \tilde{B}_r = B_r - \beta\delta_{y,r} = \bar{M}_r \cdot \bar{x} - \beta\delta_{y,r} \; .$$

To get a mistake bound for this modified version of MIRA we apply Theorem 9 almost verbatim by replacing B_r with $\tilde{B}_r$ in the theorem. Note that if $\tilde{B}_y - \max_{r \neq y}\tilde{B}_r \leq -\beta$ then $B_y - \beta - \max_{r \neq y} B_r \leq -\beta$ and hence $B_y - \max_{r \neq y} B_r \leq 0$. Therefore, for any $0 \leq \beta \leq 2R^2$ we get that the number of mistakes of the modified algorithm is equal to n_β which is bounded by $4R^4/\beta\gamma^2$. This gives the following corollary.

Corollary 10 *Let $(\bar{x}^1, y^1), \ldots, (\bar{x}^T, y^T)$ be an input sequence to the aggressive version of MIRA with margin $0 \leq \beta \leq 2R^2$, where $\bar{x}^t \in \mathbb{R}^n$ and $y^t \in \{1, 2, \ldots, k\}$. Denote by $R = \max_t \|\bar{x}^t\|$ and assume that there is a prototype matrix $\mathbf{M}^*$ of a unit vector-norm, $\|\mathbf{M}^*\|_2 = 1$, which classifies the entire sequence correctly with margin $\gamma = \min_t\{\bar{M}^*_{y^t} \cdot \bar{x}^t - \max_{r \neq y^t} \bar{M}^*_r \cdot \bar{x}^t\} > 0$. Then, the number of mistakes the algorithm makes is bounded above by,*

$$4\frac{R^4}{\beta\gamma^2} \; .$$

Note that the bound is a decreasing function of β. This means that the more aggressive we are by requiring a minimal margin the smaller the bound on the number of mistakes the aggressively modified MIRA makes. However, this also implies that the algorithm will update $\mathbf{M}$ more often and the solution will be less sparse. We conclude this section with the binary version of the aggressive algorithm. As in the multiclass setting, we replace the non-aggressive version given by

Initialize:

- Fix $\eta > 0$.

version 1

- Set $M^1_{r,i} = \frac{1}{n}$

version 2

- Set $M^1_{r,i} = \frac{1}{nk}$

Loop: $t = 1, 2, \ldots, T$

- Get a new instance $\bar{x}^t \in \mathbb{R}^n$.
- Predict $\hat{y}^t = \arg\max_{r=1}^{k} \{\bar{M}^t_r \cdot \bar{x}^t\}$.
- Get a new label y^t.
- Set $E = \{r \neq y^t \ : \bar{M}^t_r \cdot \bar{x}^t \geq \bar{M}^t_{y^t} \cdot \bar{x}^t\}$.
- If $E \neq \emptyset$ update $\mathbf{M}^t$:
  - Choose any $\tau^t_1, \ldots, \tau^t_k$ subject to :
    1. $\tau^t_r \leq \delta_{r,y^t}$ for $r = 1, \ldots, k$.
    2. $\sum_{r=1}^{k} \tau^t_r = 0$
    3. $\tau^t_r = 0$ for $r \notin E \cup \{y^t\}$.
    4. $\tau^t_{y^t} = 1$.

version 1

  - Define : $Z^t_r = \sum_i M^t_{i,r} e^{\eta \tau^t_r x^t_i}$
  - Update : $M^{t+1}_{i,r} \leftarrow \frac{1}{Z^t_r} M^t_{i,r} e^{\eta \tau^t_r x^t_i}$

version 2

  - Define : $Z^t = \sum_{i,r} M^t_{i,r} e^{\eta \tau^t_r x^t_i}$
  - Update : $M^{t+1}_{i,r} \leftarrow \frac{1}{Z^t} M^t_{i,r} e^{\eta \tau^t_r x^t_i}$

Output : $H(\bar{x}) = \arg\max_r \{\bar{M}^{T+1}_r \cdot \bar{x}\}$.

Figure 5: A family of multiclass multiplicative algorithms.

Equation (17) with the corresponding aggressive version and get,

$$\begin{aligned} \min_{\tau} \quad & Q = \|\bar{x}\|^2 \tau^2 + [2y(\bar{w} \cdot \bar{x}) - \beta]\,\tau \\ \text{subject to :} \quad & 0 \leq \tau \leq 1 \ . \end{aligned}$$

Analogously to Equation (18) the solution of the problem is given by,

$$\tau = G\left(-\frac{y^t(\bar{w} \cdot \bar{x}^t) - \frac{1}{2}\beta}{\|\bar{x}^t\|^2} \right) \ .$$

All the algorithms presented so far can be straightforwardly combined with kernel methods (Vapnik, 1998). Assume that we have determined a matrix $\mathbf{M}$ by learning the coefficients $\bar{\tau}^1, \ldots, \bar{\tau}^T$ from a sequence $\{(\bar{x}^1, y^1), \ldots, (\bar{x}^T, y^T)\}$. Formally, the rth row of $\mathbf{M}$ is,

$$\bar{M}_r = \sum_{t=1}^{T} \tau^t_r \bar{x}^t \ .$$

To use $\mathbf{M}$ for classifying new instances we compute the similarity-score of an instance $\bar{x}$ for class r by multiplying $\bar{x}$ with the rth row of $\mathbf{M}$ and get,

$$\bar{M}_r \cdot \bar{x} = \sum_{t=1}^{T} \tau^t_r \left(\bar{x}^t \cdot \bar{x} \right) \ . \tag{21}$$

As in many additive online algorithms, the value of the similarity-score is a linear combination of inner-products of the form $(\vec{x}^t \cdot \vec{x})$. We therefore can replace the inner-product in Equation (21) (and also in the algorithms outlined in Figure 2 and Figure 3) with a general inner-product kernel $K(\cdot,\cdot)$ that satisfies Mercer's conditions (Vapnik, 1998). We now obtain algorithms that work in a high dimensional space. It is also simple to incorporate voting schemes (Helmbold and Warmuth, 1995, Freund and Schapire, 1999) into the above algorithms.

Before proceeding to multiplicative algorithms, let us summarize the the results we have presented so far. We started with the Perceptron algorithm and extended it to multiclass problems. By replacing the specific update of the extended Perceptron algorithm with a relaxed set of linear constrains we obtained a whole family of ultraconservative additive algorithms. We derived a mistake bound that is common to all the algorithms in the family. We then added a constraint on the norm of the coefficients used in each update to obtain MIRA. By incorporating minimal margin requirements into MIRA we get a more robust algorithm. Finally, we closed the circle by analyzing MIRA for binary problems. The result is a Perceptron-like update with a margin dependent learning rate.

6. A Family of Multiplicative Multiclass Algorithms

We now derive a family of ultraconservative multiplicative algorithms for the multiclass setting in an analogous way to the additive family of algorithms. We give the pseudo code for the multiplicative family in Figure 5. Note that two slightly different version are described. The difference in the versions is due to the different normalization for $\mathbf{M}$. In the first version we normalize $\mathbf{M}$ after each update such that the norm of each of its rows is 1, while in the second version the vector-norm of $\mathbf{M}$ is fixed to 1. The mistake bounds of the the two versions are similar as the next theorem shows.

Theorem 11 *Let $(\vec{x}^1,y^1),\ldots,(\vec{x}^T,y^T)$ be an input sequence for either the first or the second version of the multiclass algorithm from Figure 5, where $\vec{x}^t \in \mathbb{R}^n$ and $y^t \in \{1,2,\ldots,k\}$. Assume also that for all t $\|\vec{x}^t\|_\infty \leq 1$. Assume that there is a matrix $\mathbf{M}^*$ such that either $\|\bar{M}^*_r\|_1 = 1$ for $r = 1,\ldots,k$ (first version) or $\|\mathbf{M}^*\|_1 = 1$ (second version) and that the input sequence is classified correctly with margin $\gamma = \min_t\{\bar{M}^*_{y^t} \cdot \vec{x}^t - \max_{r\neq y^t} \bar{M}^*_r \cdot \vec{x}^t\} > 0$. Then there is some $\eta > 0$ for which the number of mistakes that the algorithm makes is,*

$$O\left(\frac{k^2\log(n)}{\gamma^2}\right) ,$$

for the first version, and

$$O\left(\frac{\log(n)+\log(k)}{\gamma^2}\right) ,$$

for the second version.

To compare the bounds of the two versions we need to examine the value of the minimal margin. The first version normalizes each row separately while the second normalizes the concatenation of the rows to 1. In the first version we therefore have that for all t, $\|\bar{M}^*_r\|_1 = 1$ and thus, using our definition of vector-norms we have $\|\mathbf{M}^*\|_1 = k$. Thus, if we scale the margin in the second version so that $\|\mathbf{M}^*\|_1 = k$, the mistake bound becomes

$$O\left(k^2\frac{\log(n)+\log(k)}{\gamma^2}\right) ,$$

Initialize: Set $\mathbf{M}^1 = 0$.
Loop: For $t = 1, 2, \ldots, T$

- Get a new instance $\bar{x}^t \in \mathbb{R}^n$.
- Predict $\hat{y}^t = \arg\max_{r=1}^{k} \{\bar{M}_r^t \cdot \bar{x}^t\}$.
- Get a new label y^t.
- Set $E^t = \{r \neq y^t \ : \bar{M}_r^t \cdot \bar{x}^t \geq \bar{M}_{y^t}^t \cdot \bar{x}^t\}$.
- If $E^t \neq \emptyset$ update $\mathbf{M}^t$ (otherwise $\mathbf{M}^{t+1} = \mathbf{M}^t$) :
  - Choose any $\tau_1^t, \ldots, \tau_k^t$ which satisfy the constraints:
    1. $\tau_r^t \leq \delta_{r,y^t}$ for $r = 1, \ldots, k$.
    2. $\sum_{r=1}^k \tau_r^t = 0$
    3. $\tau_r^t = 0$ for $r \notin E^t \cup \{y^t\}$.
    4. $\tau_{y^t}^t = 1$.
  - Set $\mathbf{M}^{t+1}$ to be the solution of:

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{M}\|_2^2 \\ \text{subject to :} & (1)\ \sum_{r=1}^k \tau_r^t (\bar{M}_r \cdot \bar{x}^t) \geq 1 \\ & (2)\ \mathbf{M} \cdot \mathbf{M}^t \geq \|\mathbf{M}^t\|_2^2 \end{array} \tag{22}$$

Output : $H(\bar{x}) = \arg\max_r \{\bar{M}_r^{T+1} \cdot \bar{x}\}$.

Figure 6: A multiclass version of ROMMA.

which is larger than the mistake bound of the first version by an additive factor of $k^2 \log(k)/\gamma^2$. We prove the theorem for the first version. The proof for the second version is slightly simpler and follows the same line of proof. Since the proof of both versions are fairly mundane, the proof is deferred to Appendix A.

7. A Family of Relaxed Maximum Margin Algorithms

In this section we describe an analyze Li and Long's (2002) Relaxed Online Maximum Margin Algorithm (ROMMA) with our ultraconservative framework. The result is a third family of ultraconservative algorithms. We start with a review of the underlying ideas that motivated ROMMA and then present our related family of multiclass algorithms.

ROMMA (Li and Long, 2002) is an elegant online algorithm that employs a hyperplane which is updated after each prediction error, hence denoted $\bar{w}^t \in \mathbb{R}^n$. On round t ROMMA is fed with an instance $\bar{x}^t$ and its prediction is set to $\text{sign}(\bar{w}^t \cdot \bar{x}^t)$. In case of a prediction error, $y^t(\bar{w}^t \cdot \bar{x}^t) < 0$, ROMMA algorithm updates the weight vector $\bar{w}^t$ as follows. The new weight vector $\bar{w}^{t+1}$ is chosen such that it is the vector $\bar{w}$ which attains the minimal norm subject to the following two linear constraints. The first constraint, $y^t(\bar{w} \cdot \bar{x}^t) \geq 1$, requires that the prediction of the weight vector after the update, $\bar{w}^{t+1}$, on $\bar{x}^t$ is correct and its is at least 1, namely, $y^t(\bar{w}^{t+1} \cdot \bar{x}^t) \geq 1$. The second constraint, $\bar{w} \cdot \bar{w}^t \geq \|\bar{w}^t\|^2$, imposes, rather tacitly, that the new vector $\bar{w}^{t+1}$ classifies accurately the *previous* examples. Li and Long showed that the half-space $\{\bar{w} \ : \ \bar{w} \cdot \bar{w}^t \geq \|\bar{w}^t\|^2\}$ contains the sub-space $\cap_{i=1}^{t-1}\{y^i(\bar{w} \cdot \bar{x}^t) \geq 1\}$. Hence, the second constraint can be viewed as an approximation to the set of

constraints $y^i(\vec{x}^i \cdot \vec{w}) \geq 1$ for $i = 1, \ldots, t-1$. ROMMA is a conservative algorithm – on rounds it predicts correctly it does not not modify the weight vector and simply set $\vec{w}^{t+1} = \vec{w}^t$.

We now describe how to construct an ultraconservative family based on ROMMA. As before, the ROMMA-based algorithms maintain a prototype matrix $\mathbf{M}$. Given a new instance $\vec{x}^t$, any algorithm in the family sets the predicted label to be the index of the prototype from $\mathbf{M}$ which attains the highest similarity-score, $H(\vec{x}^t) = \arg\max_{r=1}^{k} \{\bar{M}_r^t \cdot \vec{x}^t\}$. The prototype matrix is updates only on rounds on which a prediction error was made. In such cases the new prototype matrix $\mathbf{M}^{t+1}$ is set to be the matrix $\mathbf{M}$ with minimal vector-norm under the following two linear constraints. First, we require that the new prototype-matrix classifies the instance $\vec{x}^t$ correctly with a margin of at least one, that is, $\bar{M}_{y^t} \cdot \vec{x}^t - \bar{M}_r \cdot \vec{x}^t \geq 1$ for $r \neq y^t$. These $k-1$ linear constrains replace the first constraint of ROMMA. Second, we want the new prototype-matrix to classify accurately the previous examples, thus, similarly to the second constraint of ROMMA we impose a second linear constraint $\mathbf{M} \cdot \mathbf{M}^t \geq \|\mathbf{M}^t\|^2$, where the vector inner-product between two matrices is as defined in Section 2.

The result of the generalized version is a multi-class algorithm which finds a prototype matrix of a minimal norm subject to k linear constraints in total. However, the algorithm is not necessarily ultraconservative and it is there is no simple solution to this constrained minimization problem. We therefore further approximate the constrained optimization problem by replacing the first $k-1$ linear constraints $\bar{M}_{y^t} \cdot \vec{x}^t - \bar{M}_r \cdot \vec{x}^t \geq 1$ for $r \neq y^t$, with a *single* linear constraint as follows. We pick a set of $(k-1)$ negative coefficients $\tau_1^t, \ldots, \tau_k^t$ (excluding $\tau_{y^t}^t$) which sum to -1 and define the linear constraint to be,

$$\sum_{r \neq y^t} (-\tau_r^t) \left(\bar{M}_{y^t} \cdot \vec{x}^t - \bar{M}_r \cdot \vec{x}^t\right) \geq \sum_{r \neq y^t} (-\tau_r^t) \cdot 1 = 1 \ .$$

This constraint is a convex combination of the above $k-1$ linear constraints. To further simplify the last constraint we also define $\tau_{y^t}^t = 1$ and rewrite the left hand side of the inequality,

$$\begin{aligned}
&\sum_{r \neq y^t} (-\tau_r^t) \left(\bar{M}_{y^t} \cdot \vec{x}^t - \bar{M}_r \cdot \vec{x}^t\right) = \\
&= \sum_{r \neq y^t} (-\tau_r^t) \left(\bar{M}_{y^t} \cdot \vec{x}^t\right) + \sum_{r \neq y^t} \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) \\
&= \left(\bar{M}_{y^t} \cdot \vec{x}^t\right) \sum_{r \neq y^t} (-\tau_r^t) + \sum_{r \neq y^t} \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) \\
&= \tau_{y^t}^t \left(\bar{M}_{y^t} \cdot \vec{x}^t\right) + \sum_{r \neq y^t} \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) \\
&= \sum_r \tau_r^t \left(\bar{M}_r \cdot \vec{x}^t\right) \ .
\end{aligned}$$

Finally, to ensure that the solution yields an ultraconservative update we impose another constraint on the coefficients $\bar{\tau}$. We again define the error set, $E^t = \{r \neq y^t \ : \bar{M}_r^t \cdot \vec{x}^t \geq \bar{M}_{y^t}^t \cdot \vec{x}^t\}$, to be the set of indices of the rows in $\mathbf{M}$ which achieve similarity-scores that are higher than the score of the correct label y^t. We now set τ_r^t to be zero for $r \notin E^t \cup \{y^t\}$.

The family of multiclass algorithms based on ROMMA, which we call MC-ROMMA, is described in Figure 6. We now turn to prove a mistake bound for this family by generalizing the proof techniques of Li and Long to multiclass setting. In order to prove the mistake-bound we need a couple of technical lemmas which are given below. The proofs of the lemmas generalizes the proof of the original ROMMA algorithm and are deferred to Appendix A. We then prove in Theorem 15 that MC-ROMMA is indeed ultraconservative.

Lemma 12 *Let $(\vec{x}^1,y^1),\ldots,(\vec{x}^T,y^T)$ be a separable input sequence for MC-ROMMA, where $\vec{x}^t \in \mathbb{R}^n$ and $y^t \in \{1,2,\ldots,k\}$. If MC-ROMMA made a prediction error on the t'th example $(E^t \neq \emptyset)$ then $\sum_{r=1}^k \tau_r^t \left(\bar{M}_r^{t+1} \cdot \vec{x}^t\right) = 1$.*

Lemma 13 *Let $(\vec{x}^1,y^1),\ldots,(\vec{x}^T,y^T)$ be a separable input sequence for MC-ROMMA where $\vec{x}^t \in \mathbb{R}^n$ and $y^t \in \{1,2,\ldots,k\}$. If MC-ROMMA makes a prediction error on the t'th example $(E^t \neq \emptyset)$ for $t > 1$ then $\mathbf{M}^{t+1} \cdot \mathbf{M}^t = \|\mathbf{M}^t\|^2$.*

We are now ready to state and prove the mistake bound for MC-ROMMA.

Theorem 14 *Let $(\vec{x}^1,y^1),\ldots,(\vec{x}^T,y^T)$ be an input sequence for MC-ROMMA where $\vec{x}^t \in \mathbb{R}^n$ and $y^t \in \{1,2,\ldots,k\}$. Denote by $R = \max_t \|\vec{x}^t\|$. Assume that there is a matrix $\mathbf{M}^*$ which classifies the entire sequence correctly with a margin of at least one, $\forall t = 1,\ldots,T, r \neq y^t :\ \bar{M}^*_{y^t} \cdot \vec{x}^t - \bar{M}^*_r \cdot \vec{x}^t \geq 1$. Then, the number of mistakes that MC-ROMMA makes is at most $2R^2\|\mathbf{M}^*\|^2$.*

Proof First, since $\mathbf{M}^*$ separates the data with a unit margin we have that $\mathbf{M}^* \cdot \mathbf{M}^t \geq \|\mathbf{M}^t\|^2$ for $t = 1,\ldots,T$. Second, since $\mathbf{M}^{t+1}$ attains the minimal norm in the corresponding optimization problem, we have $\|\mathbf{M}^*\| \geq \|\mathbf{M}^t\|$ for all t. Also, since $\mathbf{M}^1 = 0$ we can combine Lemma 12 with the proof of Lemma 13 and get that $\mathbf{M}^2 = \mathbf{a}^1$, i.e.

$$\bar{M}^2_r = \frac{\tau^1_r \vec{x}^1}{\|\vec{x}^1\|^2 \left[\sum_s (\tau^1_s)^2\right]} \ .$$

Computing the vector-norm of $\mathbf{M}^2$ we get,

$$\|\mathbf{M}^2\|^2 = \frac{1}{\|\vec{x}^1\|^2 \left[\sum_s (\tau^1_s)^2\right]} \ .$$

Finally, by applying Lemma 2 and the assumption that $R \geq \|\vec{x}^t\|$ we get,

$$\|\mathbf{M}^2\|^2 = \frac{1}{\|\vec{x}^1\| \sum_s (\tau^1_s)^2} \geq \frac{1}{2R^2} \ .$$

We show below that for all $t > 1$ whenever a prediction error occurred then $\|\mathbf{M}^{t+1}\|^2 \geq \|\mathbf{M}^t\|^2 + 1/(2R^2)$. This implies that if MC-ROMMA made m mistakes on the sequence of instances and labels then, $\|\mathbf{M}^{T+1}\|^2 \geq \|\mathbf{M}^1\|^2 + m/(2R^2) = m/(2R^2)$. Since $\|\mathbf{M}^{T+1}\|^2 \leq \|\mathbf{M}^*\|^2$ then, $m \leq 2\|\mathbf{M}^*\|^2 R^2$, which would complete the proof and therefore, it remains to show that $\|\mathbf{M}^{t+1}\|^2 \geq \|\mathbf{M}^t\|^2 + 1/(2R^2)$ for any round $t > 1$ on which MC-ROMMA made a prediction error.

To show that the bound on the growth of the norm $\mathbf{M}^{t+1}$ with respect to the norm of $\mathbf{M}^t$ we examine the distance $d(\mathbf{M}^t, A^t)$ between the matrix $\mathbf{M}^t$ and the set of hyperplanes $A^t = \{\mathbf{M} : \sum_r \tau^t_r (\bar{M}_r \cdot \vec{x}^t) = 1\}$ which was defined in the proof of Lemma 13. We now use the assumption that the tth example was misclassified ($\sum_r \tau^t_r (\bar{M}_r \cdot \vec{x}^t) < 0$) and Lemma 2 to get,

$$\begin{aligned} d(\mathbf{M}^t, A^t) &= \frac{\left|\sum_r \tau^t_r (\bar{M}_r \cdot \vec{x}^t) - 1\right|}{\|\vec{x}^t\| \sqrt{\sum_s (\tau^t_s)^2}} \\ &\geq \frac{1}{\sqrt{2}\|\vec{x}^t\|} \\ &\geq \frac{1}{\sqrt{2}R} \ . \end{aligned} \tag{23}$$

Also, since the new matrix $\mathbf{M}^{t+1}$ is in the set A^t then the distance between $\mathbf{M}^t$ and $\mathbf{M}^{t+1}$ is at least as big as the distance between $\mathbf{M}^t$ and A^t, that is,

$$d(\mathbf{M}^t, \mathbf{M}^{t+1}) \geq d(\mathbf{M}^t, A^t) \ . \tag{24}$$

Combining Equations (23) and (24) we get,

$$\|\mathbf{M}^{t+1} - \mathbf{M}^t\|^2 \geq \frac{1}{2R^2} \ . \tag{25}$$

We now expand the norm $\|\mathbf{M}^{t+1}\|^2$,

$$\begin{aligned} \|\mathbf{M}^{t+1}\|^2 &= \|(\mathbf{M}^{t+1} - \mathbf{M}^t) + \mathbf{M}^t\|^2 \\ &= \|\mathbf{M}^{t+1} - \mathbf{M}^t\|^2 + \|\mathbf{M}^t\|^2 - 2(\mathbf{M}^{t+1} - \mathbf{M}^t) \cdot \mathbf{M}^t \\ &= \|\mathbf{M}^{t+1} - \mathbf{M}^t\|^2 + \|\mathbf{M}^t\|^2 - 2\left(\mathbf{M}^{t+1} \cdot \mathbf{M}^t - \|\mathbf{M}^t\|^2\right) \end{aligned}$$

Using Lemma 13 we know that $\mathbf{M}^{t+1} \cdot \mathbf{M}^t - \|\mathbf{M}^t\|^2 = 0$ and thus,

$$\|\mathbf{M}^{t+1}\|^2 = \|\mathbf{M}^{t+1} - \mathbf{M}^t\|^2 + \|\mathbf{M}^t\|^2 \ . \tag{26}$$

Combining Equations (25) and (26) we get,

$$\|\mathbf{M}^{t+1}\|^2 \geq \|\mathbf{M}^t\|^2 + \frac{1}{2R^2} \ ,$$

which completes the proof. ■

Finally, we conclude this section by showing that MC-ROMMA is ultraconservative.

Theorem 15 *MC-ROMMA is ultraconservative.*

Proof We first show that the optimization problem given in Equation (22) can be re-rewritten as a constrained optimization where the unknown variables can be grouped into a single matrix in $\mathbb{R}^{n \times k}$. We replace the prototype-matrix $\mathbf{M}$ with the vector $(\bar{M}_1, \ldots, \bar{M}_k)$ and the instance $\vec{x}^t$ with the vector $(\tau_1^t \vec{x}^t, \ldots, \tau_k^t \vec{x}^t)$. It is straightforward to verify that the optimization problem of Equation (22) can now be rewritten as,

$$\begin{aligned} \min \quad & \|(\bar{M}_1, \ldots, \bar{M}_k)\|^2 \\ \text{subject to:} \quad & (\bar{M}_1, \ldots, \bar{M}_k) \cdot (\tau_1^t \vec{x}^t, \ldots, \tau_k^t \vec{x}^t) \geq 1 \\ & (\bar{M}_1, \ldots, \bar{M}_k) \cdot (\bar{M}_1^t, \ldots, \bar{M}_k^t) \geq \|(\bar{M}_1^t, \ldots, \bar{M}_k^t)\|^2 \ . \end{aligned}$$

Applying Lemma 12 and Lemma 13 we get that that the optimum of Equation (27) is achieved when the inequalities hold as equalities. The same property holds for the original version of ROMMA. We therefore can use Li and Long's closed form solution and get that the solution is of the form,

$$(\bar{M}_1^{t+1}, \ldots, \bar{M}_k^{t+1}) = c_t(\bar{M}_1^{t+1}, \ldots, \bar{M}_k^{t+1}) + d_t(\tau_1^t \vec{x}^t, \ldots, \tau_k^t \vec{x}^t) \ ,$$

for some values $c_t > 0$ and d_t. Going back to the representation that employs multiple matrices we get that the value of the prototype-matrix after the update is,

$$\forall r \quad \bar{M}_r^{t+1} = c_t \left(\bar{M}_r^t + \frac{d_t}{c_t} \tau_r^t \vec{x}^t \right) \ . \tag{27}$$

Name	No. of Training Examples	No. of Test Examples	No. of Classes	No. of Attributes
Chess-Board	10,000	10,000	8	2
MNIST	60,000	10,000	10	784
USPS	7,291	2,007	10	256
Letter	16,000	4,000	26	16

Table 1: Data sets learning problems used in the experiments

The updated given by Equation (27) can be decomposed into two stages. First, similar to the family of additive algorithms of Figure 2 and to MIRA (Figure 3), the algorithm replaces the prototype $\bar{M}_r^t$ with the sum $\bar{M}_r^t + (d_t/c_t)\tau_r^t \bar{x}^t$. Using the third condition of MC-ROMMA (Figure 6) we get that if the label r was not one of the sources for an error then $\tau_r^t = 0$ and therefore $\bar{M}_r^{t+1} = \bar{M}_r^t$. Therefore the update is ultraconservative. After the additive change to $\bar{M}_r^t$, the MC-ROMMA scales *all* the prototypes by a multiplicative factor c_t. Although all of the prototypes are modified in this stage, including those which are not in the error set ($r \notin E^t$), the classification function $H(\bar{x})$ induced by $\bar{M}_r^t$ is *not* affected by this scaling and thus the update rule is can be viewed as ultraconservative. ■

8. Experiments

In this section we describe and discuss the results of experiments we performed with both synthetic data and natural datasets. The experiments are by no means exhaustive and the main goal of these experiments is to underscore the merits of the various online algorithms discussed in this paper.

Algorithms: We compared the following five algorithms. The first algorithm is a multiclass classifier based on the Perceptron algorithm obtained by training several copies of the Perceptron. Each copy is trained to discriminate one class from the rest of the classes. To classify a new instance we compute the output of each of the trained Perceptrons and predict the label which attains the highest similarity-score. This approach can be viewed as a special case of error correcting output codes (ECOC), used for reducing a multiclass problems into multiple binary problems (Dietterich and Bakiri, 1995, Allwein et al., 2000). The next three algorithms belong to the family of algorithms discussed in Section 4 and whose pseudo-code is given in Figure 2. Each of the three algorithms corresponds to a different update. All the three algorithms replace $\bar{M}_y$ with $\bar{M}_y + \bar{x}$ whenever the prediction is incorrect. In addition each of the algorithms modify the set of prototypes constituting the error set. Specifically, the first update changes the prototypes in the error set in a uniform manner by adding the vector $-\bar{x}/|E|$ to each prototype and is thus referred to as the *uniform* update. The second update is more conservative and changes only two of the prototypes on each round: the prototype $\bar{M}_y$ corresponding to the correct label y and the prototype $\bar{M}_r$ which attains the highest similarity-score. This update is therefore referred to as *max* update. Last, the third update modifies each prototype from the error-set in proportion to the similarity-score it attains (see Section 4 for a formal description) an is abbreviated as the *prop* update. We ran all the algorithms above in an aggressive fashion: on each round a value of $\beta = 0.01$ was deducted from the similarity-score of the correct label y right before computing the error-set and the corresponding update. This modification

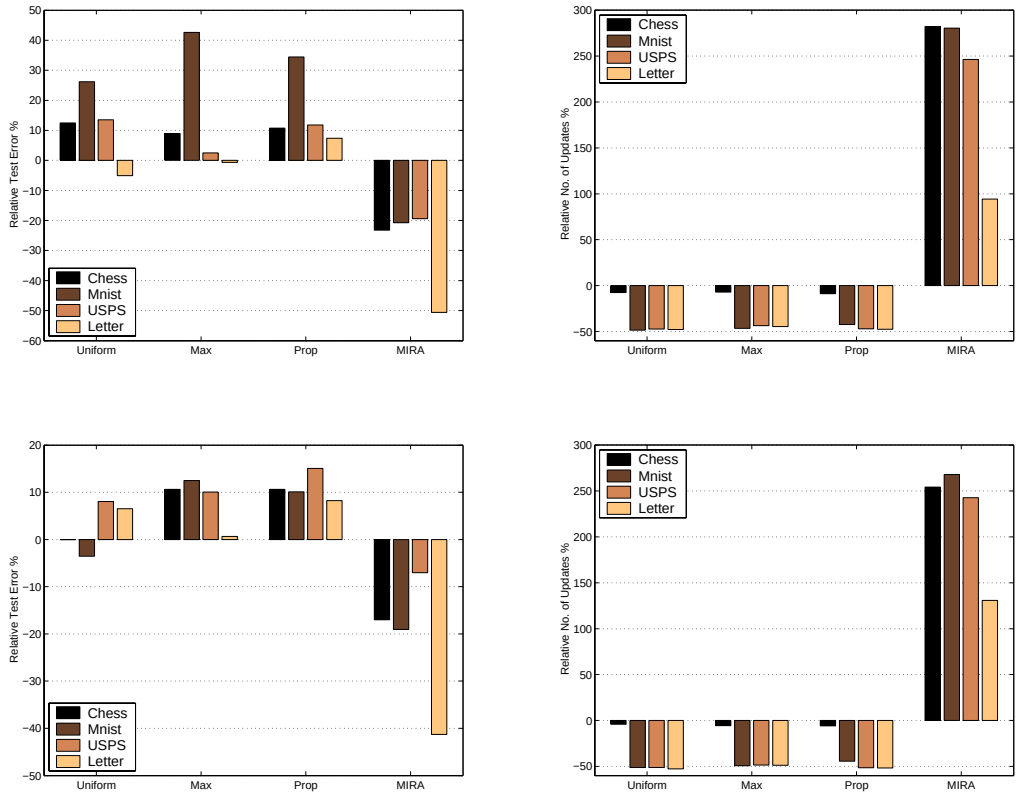

Figure 7: The relative test error (left) and relative number of updates (right) of four of the algorithms presented in this paper after one epoch (top row) and after three epochs (bottom row).

of the score forces the algorithms to perform an update even on rounds with no prediction error as long as the margin is smaller than $\beta = 0.01$. The fifth algorithm that we tested is an aggressive version of MIRA with a minimal margin requirement of $\beta = 0.01$. All of the algorithms were used in conjunction with Mercer kernels. The kernels were fixed for each dataset we experimented with and we did no attempt to tune their parameters.

Each of the five algorithms was fed with the training set in an online fashion, i.e. example by example, and generated a multiclass classification rule. We then evaluated the algorithms by applying their final set of prototypes to the test data and computed their test error. We repeated these experiments multiple times. (The specific number of repetitions varies between the datasets in is reported below.)

Data-Sets: We evaluated the algorithms on a synthetic dataset and on three natural datasets: `MNIST`[1], `USPS`[2] and `Letter`[3]. The characteristic of the sets are summarized in Table 1. A comprehensive overview of the performance of various algorithms on these sets can be found in a recent paper by Gentile (2001).

1. Available from `http://www.research.att.com/~yann/exdb/mnist/index.html`
2. Available from `ftp.kyb.tuebingen.mpg.de`
3. Available from `http://www.ics.uci.edu/~mlearn/MLRepository.html`

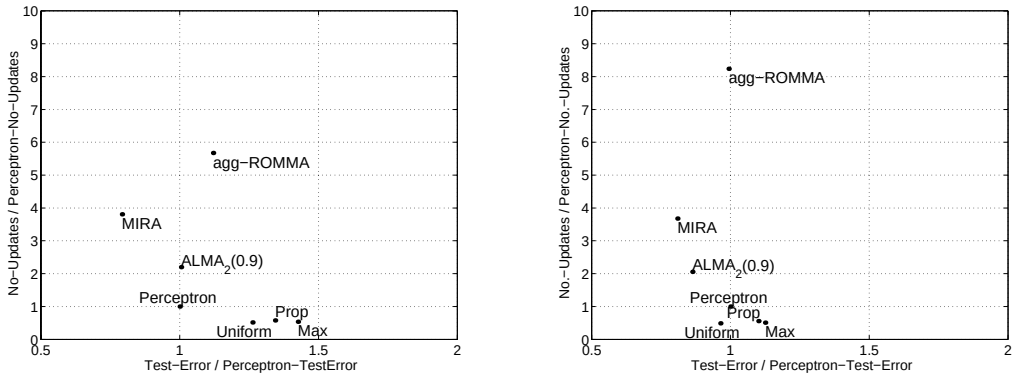

Figure 8: Summary of the test error and the number of updates for various online Please refer to the text for the exact setting used for each of the algorithms.

The synthetic data-set has eight classes. Each instances is a two dimensional vector from $[0,1] \times [0,1]$. We used the uniform distribution to randomly draw examples. Each example was associated with a unique label according to the following rule. The domain $[0,1] \times [0,1]$ was partitioned into $8 \times 8 = 64$ squares of the same size. Each square was uniquely identified with a row-column index (i, j). The label of all instances from a given square indexed (i, j) was set to be $((i+j) \bmod 8) + 1$. We then generated a training set and a test set, each of size $10,000$.

Results: The complete results obtained in the experiments are summarized in Appendix B. The appendix also cites performance results for ROMMA (Li and Long, 2002) and ALMA (Gentile, 2001). A graphical illustration that compares the algorithms described in this paper is given in Figure 7. This figure contains four bar-plots. Each bar in the plots designates corresponds to a ratio of a performance measure of one the algorithms and the Perceptron algorithm: the left two plots show the relative test error and the right two plots show the relative number of updates each algorithm performed. Formally, the height of each bar in the left two plots is proportional to $(\varepsilon_a - \varepsilon_p)/\varepsilon_p$ where ε_p is the test error of the Perceptron algorithm and ε_a is the test error on one of the other four algorithms (Uniform, Max, Prop and MIRA). Similarly, the height of each bar is the right two plots is proportional to $(u_a - u_p)/u_p$ where $u_p(u_a)$ is the number of updates the Perceptron algorithm (one of the four algorithms; Uniform, Max, Prop and MIRA) made. The top two plots refers to the results after cycling once through the training data and the bottom two plots refers to the results after three cycles through the training data. In each plots there are four groups of bars, one for each for one of the four multiclass algorithms described in this paper (Uniform, Max, Prop and MIRA). The results for each consist of four bars corresponding to four datasets: `Chess-Board`, `MNIST`, `USPS` and `Letter` (from left to right).

From the figure we see that MIRA outperforms the other algorithms described in this paper, but this improved performance has a price in terms of the sparseness of the solution. The test error of the Perceptron is lower than the test error of the rest of the algorithms (Uniform, Max, and Prop) but the Perceptron performs more updates than the three hence the resulting classifier is less sparse. For instance, for the USPS dataset, the test error of Uniform, Max, and Prop is about 10% higher than the error of the Perceptron while the test error of MIRA is around 20% lower than that of the

Perceptron. The advantage of MIRA over the Perceptron is even more evident in the `Letter` dataset where MIRA's test error is lower by 50% than the Perceptron's error. After three epochs the test error of the Uniform update becomes only 8% higher than the error of the Perceptron algorithm on three datasets and the Uniform update outperforms the Perceptron on `MNIST`. Whether one epoch or three, MIRA outperforms all of the algorithms. However, MIRA makes many more updates which results in large number of support patterns when kernel are used. The number of support patterns used by MIRA after one epoch is about four times the number used by the Perceptron (two times on the `Letter` data-set). Uniform, Max and Prop, on the other hand, makes about half of the number of updates compared to the Perceptron algorithm. This behaviour does not change after three epochs.

Another perspective of the results on the `MNIST` data-set is illustrated in Figure 8. The plot on the left hand side plot corresponds to results obtained after one epoch while the right hand side plot corresponds to results obtained after three epochs. In each of the two plots the x-axis designates the test error of an algorithm divided by the test error of the Perceptron algorithm and the y-axis is the number of updates the algorithm made divided by the number of updates of the Perceptron. Each of the algorithm is thus associate with a coordinate in each plot. By definition, the Perceptron algorithm is the point $(1,1)$. We added to the plots the results obtained by two more algorithms: Li and Long's (2002) ROMMA algorithm and Gentile's (2001) ALMA algorithm. These algorithms were designed for binary classification problems and were adapted for multiclass problems using the one-vs-rest reduction. Li and Long evaluated ROMMA on `MNIST` using a non-homogeneous polynomial kernel of degree four in an aggressive manner. ALMA was evaluated using a non-homogeneous polynomial kernel of degree six. In the experiments with these algorithm, each input instance was normalized to have an l_∞ of one. The plots appearing in Figure 8 further underscore the tradeoff between accuracy and sparseness. While MIRA exhibits the lowest error rate, with the exception of ROMMA, it is also the algorithm that makes the largest number of updates. Analogously, the three updates from Figure 2 make far less updates at the expense of inferior performance. ROMMA seems to exhibit somewhat poorer performance in terms of the accuracy versus number of updates ratio while ALMA seems to be comparable in terms of that ratio. We would like to note these performance differences might be attributed to the different pre-processing and different kernels used in our experiments. Nonetheless, all algorithms do exhibit a natural tradeoff between accuracy and sparseness of the solution.

9. Summary

In this paper we described a general framework for deriving ultraconservative algorithms for multiclass categorization problems and analyzed the proposed algorithms in the mistake bound model. We investigated in detail an additive family of online algorithms. The entire family reduces to the Perceptron algorithm in the binary case. In addition, we gave a method for choosing a unique member of the family by imposing a quadratic objective function that minimizes the norm of the prototype matrix after each update. We then gave an analogous family of multiplicative algorithms. A question that remains open is how to impose constraints similar to the one MIRA employs in the multiplicative case. We also described an ultraconservative version of Li and Long's ROMMA algorithm. We believe that the ultraconservative approach to multiclass problems can be also be applied to to quasi-additive algorithms (Grove et al., 2001) and p-norm algorithms (Gentile, 2001). Another interesting direction for research that generalizes our framework is the design and analysis of algorithms that maintain more than one prototype per class. While this approach is clearly useful

in cases where the distribution of instances from a given class is not concentrated in one direction, it seems rather tricky to generalize the ultraconservative paradigm to the case of multiple prototypes.

We would like to note that this work is part of a general line of research on multiclass learning. Allwein et al. (2000) described and analyzed a general approach for multiclass problems using error correcting output codes (Dietterich and Bakiri, 1995). Building on that work, we (Crammer and Singer, 2000) investigated the problem of designing good output codes for multiclass problems. Although the model of learning using output code differs substantially from the framework studied in this paper, a few of the techniques presented here build upon other results (Crammer and Singer, 2000). Finally, a few of the techniques used in this paper can also be applied in batch settings to construct Multiclass Support Vector Machines (MSVM). The implementation details on how to efficiently build MSVMs appear in another place (Crammer and Singer, 2001).

Acknowledgement

We would like to thank Elisheva Bonchek for carefully reading a draft of the manuscript and to Noam Slonim for useful comments. We also would like to thank the anonymous reviewers and the action editor for their constructive comments. Last, we would like to acknowledge the financial support of EU project KerMIT No. IST-2000-25341.

Appendix A. Technical Proofs

Proof of Theorem 4:
The case $D = 0$ follows from Theorem 3 thus we can assume that $D > 0$. The theorem is proved by transforming the non-separable setting to a separable one. To do so, we extend each instance $\vec{x}^t \in \mathbb{R}^n$ to $\vec{z}^t \in \mathbb{R}^{n+T}$ as follows. The first n coordinates of $\vec{z}^t$ are set to $\vec{x}^t$. The $n+t$ coordinate of $\vec{z}^t$ is set to Δ, which is a positive real number whose value is determined later; the rest of the coordinates of $\vec{z}^t$ are set to zero. We similarly extend the matrix $\mathbf{M}^*$ to $\mathbf{W}^* \in \mathbb{R}^{k\times(n+T)}$ as follows. We set the first n columns $\mathbf{W}^*$ to be $\frac{1}{Z}\mathbf{M}^*$. For each row r we set $W^*_{r,n+t}$ to $\frac{d^t}{Z\Delta}$ if $r = y^t$ and zero otherwise. To summarize, the structure of $\mathbf{W}^*$ is,

$$\mathbf{W}^* = \frac{1}{Z}\left[\begin{array}{c|c} \mathbf{M}^* & \delta_{r,y^t}\frac{d^t}{\Delta} \end{array}\right] .$$

We choose the value of Z so that $\|\mathbf{W}^*\|_2 = 1$, hence,

$$1 = \|\mathbf{W}^*\|_2^2 = \frac{1}{Z^2}\left(1 + \frac{D^2}{\Delta^2}\right)$$

which gives that,

$$Z = \sqrt{1 + \frac{D^2}{\Delta^2}} .$$

We now show that $\mathbf{W}^*$ achieves a margin of $\frac{\gamma}{Z}$ on the extended data sequence. Note that for all r and t,

$$\begin{aligned} \bar{W}^*_r \cdot \vec{z}^t &= \frac{1}{Z}\left(\bar{M}^*_r \cdot \vec{x}^t + \delta_{r,y^t}\frac{d^t}{\Delta}\Delta\right) \\ &= \frac{1}{Z}\left(\bar{M}^*_r \cdot \vec{x}^t + \delta_{r,y^t} d^t\right) . \end{aligned}$$

Now, using the definition of d^t we get,

$$\begin{aligned}
\bar{W}^*_{y^t} \cdot \vec{z}^t - \max_{r \neq y^t} \left\{ \bar{W}^*_r \cdot \vec{z}^t \right\} &= \frac{1}{Z} \left(\bar{M}^*_{y^t} \cdot \vec{x}^t + d^t \right) - \max_{r \neq y^t} \left\{ \frac{1}{Z} \left(\bar{M}^*_r \cdot \vec{x}^t \right) \right\} \\
&= \frac{1}{Z} d^t + \frac{1}{Z} \left[\bar{M}^*_{y^t} \cdot \vec{x}^t - \max_{r \neq y^t} \left\{ \bar{M}^*_r \cdot \vec{x}^t \right\} \right] \\
&\geq \frac{1}{Z} \left(\gamma - \left[\bar{M}^*_{y^t} \cdot \vec{x}^t - \max_{r \neq y^t} \left\{ \bar{M}^*_r \cdot \vec{x}^t \right\} \right] \right) \\
&\quad + \frac{1}{Z} \left[\bar{M}^*_{y^t} \cdot \vec{x}^t - \max_{r \neq y^t} \left\{ \bar{M}^*_r \cdot \vec{x}^t \right\} \right] \\
&= \frac{\gamma}{Z} \,. \qquad (28)
\end{aligned}$$

We also have that,

$$\|\vec{z}^t\|^2 = \|\vec{x}^t\|^2 + \Delta^2 \leq R^2 + \Delta^2 \,. \qquad (29)$$

In summary, Equation (28) and Equation (29) imply that the sequence $(\vec{z}^1, y^1), \ldots, (\vec{z}^T, y^T)$ is classified correctly with margin $\frac{\gamma}{Z}$ and each instance $\vec{z}^t$ is bounded above by $R^2 + \Delta^2$. Thus, we can use Theorem 3 and conclude that the number of mistakes that the algorithm makes on $(\vec{z}^1, y^1), \ldots, (\vec{z}^T, y^T)$ is bounded from above by,

$$2 \frac{R^2 + \Delta^2}{\left(\frac{\gamma}{Z}\right)^2} \,. \qquad (30)$$

Minimizing Equation (30) over Δ we get that the optimal value for Δ is $\sqrt{DR}$ and the tightest mistake bound is,

$$2 \frac{(D+R)^2}{\gamma^2} \,.$$

To complete the proof we show that the prediction of the algorithm in the extended space and in the original space are equal. Namely, let $\mathbf{M}^t$ and $\mathbf{W}^t$ be the value of the parameter matrix just before receiving $\vec{x}^t$ and $\vec{z}^t$, respectively. We need to show that the following conditions hold for $t = 1, \ldots, T$:

1. The first n columns of $\mathbf{W}^t$ are equal to $\mathbf{M}^t$.
2. The $(n+t)$th column of $\mathbf{W}^t$ is equal zero.
3. $\bar{M}^t_r \cdot \vec{x}^t = \bar{W}^t_r \cdot \vec{z}^t$ for $r = 1, \ldots, k$.

The proof of these conditions is straightforward by induction on t. ■

Proof of Theorem 7:
Let $\mathbf{M}$ be the prototype matrix just before round t and denote by $\mathbf{M}'$ the updated matrix after round t, that is,

$$\bar{M}'_r = \bar{M}_r + \tau^t_r \vec{x}^t \ (r = 1, 2, \ldots, k) \,.$$

As in Theorem 3, we bound $\|\mathbf{M}\|_2^2$ from above and below. First, we develop the lower bound on $\|\mathbf{M}\|_2^2$ by bounding the term,

$$\begin{aligned}\sum_{r=1}^{k} \bar{M}_r^* \cdot \bar{M}_r' &= \sum_{r=1}^{k} \bar{M}_r^* \cdot (\bar{M}_r + \tau_r^t \vec{x}^t) \\ &= \sum_{r=1}^{k} \bar{M}_r^* \cdot \bar{M}_r + \sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t\right) . \end{aligned} \tag{31}$$

We further develop the second term using the second constraint of MIRA. Substituting $\tau_{y^t} = -\sum_{r \neq y^t} \tau_r^t$ we get,

$$\begin{aligned}\sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t\right) &= \sum_{r \neq y^t} \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t\right) + \tau_{y^t} \left(\bar{M}_{y^t}^* \cdot \vec{x}^t\right) \\ &= \sum_{r \neq y^t} \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t\right) - \sum_{r \neq y^t} \tau_r^t \left(\bar{M}_{y^t}^* \cdot \vec{x}^t\right) \\ &= \sum_{r \neq y^t} \left(-\tau_r^t\right) \left(\bar{M}_{y^t}^* - \bar{M}_r^*\right) \cdot \vec{x}^t . \end{aligned}$$

Using the fact that $\mathbf{M}^*$ classifies all the instances with margin γ we obtain,

$$\sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t\right) \geq \sum_{r \neq y^t} \left(-\tau_r^t\right) \gamma = \tau_{y^t}^t \gamma . \tag{32}$$

Combining Equation (31) and Equation (32) we get,

$$\sum_r \bar{M}_r^* \cdot \bar{M}_r' \geq \sum_r \bar{M}_r^* \cdot \bar{M}_r + \tau_r^t \gamma .$$

Thus, after T rounds the matrix $\mathbf{M}$ satisfies,

$$\sum_r \bar{M}_r^* \cdot \bar{M}_r \geq \gamma \sum_t \tau_r^t . \tag{33}$$

Using the definition of the vector-norm and applying the Cauchy-Schwartz inequality we get,

$$\begin{aligned}\|\mathbf{M}\|^2 \|\mathbf{M}^*\|^2 &= \left(\sum_{r=1}^{k} \|\bar{M}_r\|^2\right) \left(\sum_{r=1}^{k} \|\bar{M}_r^*\|^2\right) \\ &\geq \left(\bar{M}_1 \cdot \bar{M}_1^* + \ldots + \bar{M}_k \cdot \bar{M}_k^*\right)^2 \\ &= \left(\sum_{r=1}^{k} \bar{M}_r \cdot \bar{M}_r^*\right)^2 . \end{aligned} \tag{34}$$

Plugging Equation (33) into Equation (34) and using the assumption that $\mathbf{M}^*$ is of a unit vector-norm we get the following lower bound,

$$\|\mathbf{M}\|^2 \geq \gamma^2 \left(\sum_t \tau_{y^t}^t\right)^2 . \tag{35}$$

Next, we bound the vector-norm of $\mathbf{M}$ from above,

$$\begin{aligned}
\|\mathbf{M}^{'}\|^2 &= \sum_r \|\bar{M}_r^{'}\|^2 \\
&= \sum_r \|\bar{M}_r + \tau_r^t \bar{x}^t\|^2 \\
&= \sum_r \|\bar{M}_r\|^2 + 2\sum_r \tau_r^t \left(\bar{M}_r \cdot \bar{x}^t\right) + \sum_r \|\tau_r^t \bar{x}^t\|^2 \\
&= \|\mathbf{M}\|^2 + 2\sum_r \tau_r^t \left(\bar{M}_r \cdot \bar{x}^t\right) + \|\bar{x}^t\|^2 \sum_r (\tau_r^t)^2 \ . \qquad (36)
\end{aligned}$$

Using the definition of MIRA (Figure 3) we know that $\bar{\tau}^t$ are chosen to minimize $\|\mathbf{M}^{'}\|^2$. Note that $\bar{\tau} = 0$ satisfies the constraints of MIRA and then $\mathbf{M}^{'}$ reduces to $\mathbf{M}$. Therefore we have that,

$$2\sum_r \tau_r^t \left(\bar{M}_r \cdot \bar{x}^t\right) + \|\bar{x}^t\|^2 \sum_r (\tau_r^t)^2 \leq 0 \ .$$

But $\|\bar{x}^t\|^2 \sum_r (\tau_r^t)^2 > 0$ and finally we get,

$$\sum_r \tau_r^t \left(\bar{M}_r \cdot \bar{x}^t\right) \leq 0 \ . \qquad (37)$$

Plugging Equation (37) into Equation (36) while using the bound $\|\bar{x}^t\|^2 \leq R^2$ and Lemma 2 we obtain,

$$\begin{aligned}
\|\mathbf{M}^{'}\|^2 &\leq \|\mathbf{M}\|^2 + 2\|R\|^2 \left(\tau_{y^t}^t\right)^2 \\
&\leq \|\mathbf{M}\|^2 + 2\|R\|^2 \tau_{y^t}^t \ . \qquad (38)
\end{aligned}$$

Thus, after T round the matrix $\mathbf{M}$ satisfies,

$$\|\mathbf{M}\|^2 \leq 2\|R\|^2 \sum_t \tau_{y^t}^t \ . \qquad (39)$$

Combining Equation (35) and Equation (39) we obtain,

$$\gamma^2 \left(\sum_t \tau_{y^t}^t\right)^2 \leq \|\mathbf{M}\|^2 \leq 2\|R\|^2 \sum_t \tau_{y^t}^t$$

and therefore,

$$\sum_t \tau_{y^t}^t \leq 2\frac{R^2}{\gamma^2} \ .$$

Using the second constraint of the algorithm we get,

$$\|\bar{\tau}^t\|_1 = \sum_r |\tau_r^t r| = -\sum_{r \neq y^t} \tau_r^t r + \tau_r^t y^t = 2\tau_{y^t} \ ,$$

and therefore,

$$\sum_t \|\bar{\tau}^t\|_1 \leq 4\frac{R^2}{\gamma^2} \ .$$

■

Proof of Theorem 11:
Let

$$\Phi_t = \sum_{r=1}^{k} D_{kl}(\bar{M}_r^* \| \bar{M}_r^t) \ ,$$

and define $\Delta_t = \Phi_{t+1} - \Phi_t$. Note that these definitions imply that,

$$\begin{aligned}
\Delta_t &= \Phi_{t+1} - \Phi_t \\
&= \sum_r \left[\sum_i M_{r,i}^* \log\left(\frac{M_{r,i}^*}{M_{r,i}^{t+1}} \right) \right] - \sum_r \left[\sum_i M_{r,i}^* \log\left(\frac{M_{r,i}^*}{M_{r,i}^{t}} \right) \right] \\
&= \sum_r \left[\sum_i M_{r,i}^* \log\left(\frac{M_{r,i}^{t}}{M_{r,i}^{t+1}} \right) \right] \ .
\end{aligned}$$

Recall that if no error was made on the tth example ($\hat{y}^t = y^t$) then $\overline{\tau}^t = 0$, $\mathbf{M}^{t+1} = \mathbf{M}^t$ and $\Delta_t = 0$. We therefore further develop the expression for Δ_t for the case when there was a prediction error on round t,

$$\begin{aligned}
\Delta_t &= \sum_r \left[\sum_i M_{r,i}^* \log\left(\frac{Z_r^t}{e^{\eta \tau_r^t x_i^t}} \right) \right] \\
&= \sum_r \left[\log(Z_r^t) \sum_i M_{r,i}^* - \sum_i M_i^* \eta \tau_r^t x_i^t \right] \\
&= \sum_r \left[\log(Z_r^t) \|\bar{M}_r^*\|_1 - \eta \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t \right) \right] \\
&= \sum_r \left(\log(Z_r^t) \|\bar{M}_r^*\|_1 \right) - \eta \sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t \right) \ .
\end{aligned}$$

Using the assumption $\|\bar{M}_r^*\|_1 = 1$ for all $r = 1, \ldots, k$ we get,

$$\Delta_t = \sum_r \log(Z_r^t) - \eta \sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t \right) \ . \tag{40}$$

Let us now further develop both terms of the expression above. For the right term we use the second constraint of the algorithm and substitute $\tau_{y^t} = -\sum_{r \neq y^t} \tau_r^t$ to get that,

$$\sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t \right) = \sum_{r \neq y^t} (-\tau_r^t) \left(\bar{M}_{y^t}^* - \bar{M}_r^* \right) \cdot \vec{x}^t \ .$$

Using the assumption that $\mathbf{M}^*$ classifies all the instances with margin γ and the fourth constraint of the algorithm $\left(\tau_{y^t}^t = 1 \right)$ we obtain,

$$\sum_r \tau_r^t \left(\bar{M}_r^* \cdot \vec{x}^t \right) \geq \sum_{r \neq y^t} (-\tau_r^t) \gamma = \gamma \tau_{y^t}^t = \gamma \ . \tag{41}$$

To bound the left term we use the inequality :

$$\forall \eta > 0 \ , \ x \in [-1, 1] \qquad e^{\eta x} \leq \frac{1+x}{2} e^{\eta} + \frac{1-x}{2} e^{-\eta} \ .$$

Since $|\tau_r^t| \leq 1$ and $\|\vec{x}^t\|_\infty \leq 1$ then $|\tau_r^t x_i^t| \leq 1$ and thus,

$$\begin{aligned}
Z_r^t &= \sum_i M_{r,i}^t e^{\eta \tau_r^t x_i^t} \\
&\leq \sum_i M_{r,i}^t \left[\frac{1+\tau_r^t x_i^t}{2} e^{\eta} + \frac{1-\tau_r^t x_i^t}{2} e^{-\eta} \right] \\
&= \sum_i M_{r,i}^t \frac{e^{\eta}+e^{-\eta}}{2} + \sum_i M_{r,i}^t \frac{e^{\eta}-e^{-\eta}}{2} \tau_r^t x_i^t \\
&= \frac{e^{\eta}+e^{-\eta}}{2} \sum_i M_{r,i}^t + \frac{e^{\eta}-e^{-\eta}}{2} \tau_r^t \left(\bar{M}_r^t \cdot \vec{x}^t \right) \\
&= \frac{e^{\eta}+e^{-\eta}}{2} \|\bar{M}_r^t\|_1 + \frac{e^{\eta}-e^{-\eta}}{2} (-\tau_r^t) \left(\bar{M}_{y^t}^t - \bar{M}_r^t \right) \cdot \vec{x}^t + \frac{e^{\eta}-e^{-\eta}}{2} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) .
\end{aligned}$$

Note that $\|\bar{M}_r^t\|_1 = 1$ since the algorithm normalizes the rows of the matrix on every step. We assumed that there is an error in classifying $\vec{x}^t$ and, as in the additive family of algorithms, we need to consider two cases. The first case is when the label r was not the source of the error, that is $(\bar{M}_{y^t} - \bar{M}_r) \cdot \vec{x}^t > 0$. Then by using the third constraint of the algorithm we get that $\tau_r^t = 0$ and thus $(-\tau_r^t) \left(\bar{M}_{y^t} - \bar{M}_r \right) \cdot \vec{x}^t = 0$. In the second case, if the label r was a possible source of error, then $(\bar{M}_{y^t} - \bar{M}_r) \cdot \vec{x}^t \leq 0$. Using the first constraint of the algorithm we know that $\tau_r^t \leq 0$ and thus $(-\tau_r^t) \left(\bar{M}_{y^t} - \bar{M}_r \right) \cdot \vec{x}^t \leq 0$. Since $\eta > 0$ we have that $\frac{1}{2}(e^{\eta} - e^{-\eta}) > 0$ and therefore we get,

$$Z_r^t \leq \frac{e^{\eta}+e^{-\eta}}{2} + \frac{e^{\eta}-e^{-\eta}}{2} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) . \tag{42}$$

Taking the log of Equation (42) we get,

$$\begin{aligned}
\log(Z_r^t) &\leq \log \left[\frac{e^{\eta}+e^{-\eta}}{2} + \frac{e^{\eta}-e^{-\eta}}{2} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) \right] \\
&= \log \left[\frac{e^{\eta}+e^{-\eta}}{2} \left(1 + \frac{e^{\eta}-e^{-\eta}}{e^{\eta}+e^{-\eta}} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) \right) \right] \\
&= \log \left(\frac{e^{\eta}+e^{-\eta}}{2} \right) + \log \left[1 + \frac{e^{\eta}-e^{-\eta}}{e^{\eta}+e^{-\eta}} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) \right] .
\end{aligned}$$

We use the fact the $\log(x)$ is concave and therefore $\log(1+x) \leq x$ for $x \geq -1$. Since $|\tau_r^t| \leq 1$, $\|\bar{M}_{y^t}^t\|_1 = 1$, $\|\vec{x}^t\|_\infty \leq 1$ and

$$\left| \frac{e^{\eta}-e^{-\eta}}{e^{\eta}+e^{-\eta}} \right| \leq 1 ,$$

we conclude that,

$$\log(Z_r^t) \leq \log \left(\frac{e^{\eta}+e^{-\eta}}{2} \right) + \frac{e^{\eta}-e^{-\eta}}{e^{\eta}+e^{-\eta}} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) . \tag{43}$$

Plugging Equations (41) and (43) into Equation (40) we get that if there is an error on the tth instance then

$$\begin{aligned}
\Delta_t &\leq \sum_r \log \left(\frac{e^{\eta}+e^{-\eta}}{2} \right) + \sum_r \left[\frac{e^{\eta}-e^{-\eta}}{e^{\eta}+e^{-\eta}} \tau_r^t \left(\bar{M}_{y^t}^t \cdot \vec{x}^t \right) \right] - \eta\gamma \\
&= k \log \left(\frac{e^{\eta}+e^{-\eta}}{2} \right) + \frac{e^{\eta}-e^{-\eta}}{e^{\eta}+e^{-\eta}} \left(\bar{M}_y^t \cdot \vec{x}^t \right) \sum_r \tau_r^t - \eta\gamma .
\end{aligned}$$

Using the second constraint of the algorithm ($\sum_r \tau_r^t = 0$) we obtain,

$$\Delta_t \leq k \log\left(\frac{e^{\eta} + e^{-\eta}}{2}\right) - \eta\gamma \,.$$

Therefore, if the algorithm makes m mistakes on the sequence $(\vec{x}^1, y^1), \ldots, (\vec{x}^T, y^T)$ then

$$\sum_{t=1}^{T} \Delta_t \leq m \left[k \log\left(\frac{e^{\eta} + e^{-\eta}}{2}\right) - \eta\gamma \right] \,. \tag{44}$$

On the other hand,

$$\begin{aligned} \sum_{t=1}^{T} \Delta_t &= \sum_{t=1}^{T} (\Phi_{t+1} - \Phi_t) = \Phi_{T+1} - \Phi_1 \\ &\geq -\Phi_1 = -k \log(n) \,. \end{aligned} \tag{45}$$

Combining Equations (44) and Equations (45) we obtain,

$$m \left[k \log\left(\frac{e^{\eta} + e^{-\eta}}{2}\right) - \eta\gamma \right] \geq -k \log(n) \,.$$

Solving for m we get,

$$m \leq \frac{\log(n)}{\eta \frac{\gamma}{k} + \log\left(\frac{2}{e^{\eta} + e^{-\eta}}\right)} \,.$$

Minimizing over η we obtain the required bound,

$$O\left(k^2 \frac{\log(n)}{\gamma^2}\right) \,.$$

■

Proof of Lemma 12:

Note that the claim implies that the first inequality constraint of MC-ROMMA's optimization problem is satisfied with equality after the update. Assume, by contradiction that this is not the case. That is, after an update we get,

$$\sum_r \tau_r^t \left(\bar{M}_r^{t+1} \cdot \vec{x}^t\right) > 1 \,. \tag{46}$$

We now show that there exists a matrix $\mathbf{M}'$ which satisfies the constraints of the optimization problem, but achieves a norm which is smaller than the norm of $\mathbf{M}^{t+1}$. This yields a contradiction to the assumption that $\mathbf{M}^{t+1}$ is the optimal solution.

Since $\vec{x}^t$ was misclassified we need to consider the following two cases for each label r. The first case is when the label r was not the source of the error, that is $(\bar{M}_{y^t}^t - \bar{M}_r^t) \cdot \vec{x}^t > 0$. Then, using the third constraint ($r \notin E^t \cup \{y^t\} \Rightarrow \tau_r^t = 0$) we get that $\tau_r^t = 0$ and thus $(-\tau_r^t)\left(\bar{M}_{y^t}^t - \bar{M}_r^t\right) \cdot \vec{x}^t = 0$. The second case is when one of the sources of error was the label r, i.e. $(\bar{M}_{y^t}^t - \bar{M}_r^t) \cdot \vec{x}^t \leq 0$. From

the first constraint of the algorithm we know that $\tau_r^t \leq 0$ and thus $(-\tau_r^t)\left(\bar{M}_{y^t}^t - \bar{M}_r^t\right)\cdot\bar{x}^t \leq 0$. Finally, summing over all r we get,

$$\sum_{r\neq y^t}(-\tau_r^t)\left(\bar{M}_{y^t}^t - \bar{M}_r^t\right)\cdot\bar{x}^t \leq 0\ . \tag{47}$$

We further develop the left hand-side of the above equality using the second constraint of the algorithm ($\sum_r \tau_r^t = 0$) and get,

$$\begin{aligned}\sum_{r\neq y^t}\left(-\tau_r^t\right)\left(\bar{M}_{y^t}^t - \bar{M}_r^t\right)\cdot\bar{x}^t &= \sum_{r\neq y^t}\tau_r^t\left(\bar{M}_r^t\cdot\bar{x}^t\right) - \sum_{r\neq y^t}\tau_r^t\left(\bar{M}_{y^t}^t\cdot\bar{x}^t\right)\\ &= \sum_{r\neq y^t}\tau_r^t\left(\bar{M}_r^t\cdot\bar{x}^t\right) + \tau_{y^t}\left(\bar{M}_{y^t}^t\cdot\bar{x}^t\right)\\ &= \sum_r \tau_r^t\left(\bar{M}_r^t\cdot\bar{x}^t\right)\ .\end{aligned} \tag{48}$$

Combining Equations (47) and (48) we get,

$$\sum_r \tau_r^t\left(\bar{M}_r^t\cdot\bar{x}^t\right) \leq 0\ . \tag{49}$$

From Equations (46) and (49) we get that there exists $\alpha\in(0,1)$ and $\mathbf{M}^{'} = \alpha\mathbf{M}^t + (1-\alpha)\mathbf{M}^{t+1}$ such that $\mathbf{M}^{'}$ satisfies the first constraint of the algorithm with equality, i.e. $\sum_r \tau_r^t\left(\bar{M}_r^{'}\cdot\bar{x}^t\right) = 1$. Using the definition of $\mathbf{M}^{'}$ and the convexity of the squared L_2 norm we get that,

$$\|\mathbf{M}^{'}\|^2 \leq \alpha\|\mathbf{M}^t\|^2 + (1-\alpha)\|\mathbf{M}^{t+1}\|^2\ . \tag{50}$$

Note that $\mathbf{M}^t$ is the optimal solution of the quadratic optimization problem if we omit the first inequality constraint given in Equation (22). In addition, $\mathbf{M}^t$ does not satisfy that first constraint, therefore $\|\mathbf{M}^t\|^2 < \|\mathbf{M}^{t+1}\|^2$. Plugging this inequality into Equation (50) we get,

$$\|\mathbf{M}^{'}\|^2 < \|\mathbf{M}^{t+1}\|^2\ .$$

Since both $\mathbf{M}^t$ and $\mathbf{M}^{t+1}$ satisfy the second inequality constraint of Equation (22) and $\mathbf{M}^{'}$ is a convex combination of $\mathbf{M}^t$ and $\mathbf{M}^{t+1}$, then $\mathbf{M}^{'}$ also satisfies the second constraint. Therefore, $\mathbf{M}^{'}$ is a feasible point and thus we get a contradiction. ■

Proof of Lemma 13:

Let A^t denote the set of all matrices which satisfy the first constraint with equality, that is,

$$A^t = \left\{\mathbf{M} : \sum_r \tau_r^t\left(\bar{M}_r\cdot\bar{x}^t\right) = 1\right\}\ .$$

From Lemma 12 we know that $\mathbf{M}^{t+1}\in A^t$. Define

$$\bar{a}_r^t = \frac{\tau_r^t\bar{x}^t}{\|\bar{x}^t\|^2\left[\sum_s(\tau_s^t)^2\right]}\ ,$$

and let,

$$\mathbf{a}^t = \begin{pmatrix}\bar{a}_1^t\\ \vdots\\ \bar{a}_k^t\end{pmatrix}$$

be the matrix whose rth row is $\bar{a}_r^t$. It is straightforward to verify that $\mathbf{a}^t \in A^t$. We now show that it attains the minimal vector-norm among all of the matrices in A^t. From the definitions above the norm of $\mathbf{a}^t$ is,

$$\begin{aligned}
\mathbf{a}^t \cdot \mathbf{a}^t &= \sum_r \frac{\tau_r^t \bar{x}^t}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \cdot \frac{\tau_r^t \bar{x}^t}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \\
&= \frac{\|\bar{x}^t\|^2 \left[\sum_r ({\tau_r^t}^2)\right]}{\left[\|\bar{x}^t\|^2 \sum_s (\tau_s^t)^2\right]^2} \\
&= \frac{1}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \,.
\end{aligned}$$

Also note that for every $\mathbf{M} \in A^t$ we have,

$$\begin{aligned}
\mathbf{M} \cdot \mathbf{a}^t &= \sum_r \bar{M}_r \cdot \bar{a}_r^t \\
&= \sum_r \bar{M}_r \cdot \frac{\tau_r^t \bar{x}^t}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \\
&= \frac{1}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \sum_r \tau_r^t \left(\bar{M}_r \cdot \bar{x}^t\right) \\
&= \frac{1}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \,,
\end{aligned}$$

where for the last equation we used the fact that $M \in A^t$. Combining the last two equalities we get that for all $\mathbf{M} \in A^t$,

$$\begin{aligned}
\|\mathbf{M}\|^2 &= \|(\mathbf{M} - \mathbf{a}^t) + \mathbf{a}^t\|^2 \\
&= \|\mathbf{M} - \mathbf{a}^t\|^2 + \|\mathbf{a}^t\|^2 + 2\left(\mathbf{M} \cdot \mathbf{a}^t - \mathbf{a}^t \cdot \mathbf{a}^t\right) \\
&= \|\mathbf{M} - \mathbf{a}^t\|^2 + \|\mathbf{a}^t\|^2 + 2\left(\frac{1}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} - \frac{1}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]}\right) \\
&= \|\mathbf{M} - \mathbf{a}^t\|^2 + \|\mathbf{a}^t\|^2 \,. \qquad (51)
\end{aligned}$$

Since the term on the right side of Equation (51) is constant, the norm of $\mathbf{M}$ is minimized when the term on the left hand side equals zero, that is $\mathbf{M} = \mathbf{a}^t$. However, $\mathbf{M}^{t+1} \in A^t$ and it attains the minimal norm. We therefore get $\mathbf{M}^{t+1} = \mathbf{a}^t$. We now assume by contradiction that the second inequality constraint of the optimization problem does not hold with equality for $\mathbf{M}^{t+1}$, that is $\mathbf{M}^{t+1} \cdot \mathbf{M}^t > \|\mathbf{M}^t\|^2$. Plugging the value of $\mathbf{M}^{t+1} = \mathbf{a}^t$ into the inequality we get,

$$\sum_r \frac{\tau_r^t \bar{x}^t}{\|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right]} \cdot \mathbf{M}^t > \|\mathbf{M}^t\|^2 \,.$$

Rearranging the terms we finally get,

$$\sum_r \tau_r^t \left(\bar{x}^t \cdot \mathbf{M}^t\right) > \|\mathbf{M}^t\|^2 \|\bar{x}^t\|^2 \left[\sum_s (\tau_s^t)^2\right] \,.$$

However, $\mathbf{M}^t \neq 0$ (since $t > 1$), $\vec{x}^t \neq 0$ (since the input sequence is separable) and $\sum_s (\tau_s^t)^2 > 0$ (since $E^t \neq \emptyset$), therefore,

$$\sum_r \tau_r^t \left(\vec{x}^t \cdot \mathbf{M}^t \right) > 0 \ ,$$

which is a contradiction to the assumption that there was a prediction error on round t. ■

Appendix B. Summary of Experimental Results

The results of the experiments are summarized in Tables 2 through 5. Each table contains results for a different dataset. The datasets are `Chess-Board`, `MNIST`, `USPS` and `Letter`. Each column gives results after a single pass through the training set. Each row in the tables corresponds to a specific algorithm. The top row in each pair of rows corresponds to the test error while the bottom row gives the cumulative number of updates each algorithm made. Some of the tables also contain results for ALMA (Gentile, 2001) and ROMMA (Li and Long, 2002). Both algorithms used the one-vs-rest reduction of multiclass to binary. ROMMA was trained using a non-homogeneous polynomial kernel of degree four and the data was normalized to have an l_∞ norm of 1. See (Li and Long, 2002) for further details. ALMA was designed and analyzed by Gentile (2001). ALMA was trained using different kernels than in this paper, On the `MNIST` data-set is was trained using a non-homogeneous polynomial kernel of degree six and the data was normalized to have an l_∞ norm of 1. On the `USPS` data-set is was trained using a Gaussian kernel with a standard deviation of 3.5 and on the `Letter` dataset is was trained using a ploy-Gaussian kernel. Further details are provides by Gentile (2001).

We used the prediction the last set of prototypes each algorithm outputs after cycling through the training set. However, Gentile (2001) reports that better results can be obtained by combining ALMA with a voting technique (Freund and Schapire, 1999). In the tables below we report results that were obtained without any voting or averaging techniques.

	Epochs				
Algorithm	1	2	3	4	5
Perceptron	5.6	4.9	4.7	4.7	4.6
	1891	2029	2050	2059	2062
Uniform	6.3	5.1	4.7	4.7	4.7
	1745	1933	1966	1971	1973
Max	6.1	5.4	5.2	5.1	5.1
	1758	1912	1936	1944	1947
Prop	6.2	5.3	5.2	5.1	5.1
	1723	1900	1927	1934	1938
MIRA	4.3	4.0	3.9	4.0	4.0
	7229	7259	7260	7261	7261

Table 2: Experimental results for Chess-Board data. The test error (top) and number of support patterns (bottom) for five multiclass online algorithms after $j = 1, \ldots, 10$ epochs of training on 10,000 examples.

Algorithm	Epochs 1	2	3	Kernel
Perceptron	1.83	1.58	1.68	Homogeneous
	5299	6633	7112	Polynomial
agg-ROMMA	2.05	1.76	1.67	
	30088	44495	58583	Non-Homogeneous
$ALMA_2(0.9)$	1.84	1.53	1.45	Polynomial
	11652	13712	14598	
Uniform	2.31	1.89	1.62	
	2726	3271	3458	
Max	2.61	2.13	1.89	Homogeneous
	2823	3423	3605	Polynomial
Prop	2.46	2.04	1.85	
	3050	3722	3957	
MIRA	1.45	1.37	1.36	
	20162	23878	26176	

Table 3: Experimental results for the MNIST data-set. The test error (top) and number of support patterns (bottom) for five multiclass online algorithms after $j = 1,\ldots,3$ epochs.

Algorithm	Epochs 1	2	3	4	5	Kernel
Perceptron	5.93	5.63	4.98	4.78	4.83	Homogeneous
	936	1167	1240	1266	1281	Polynomial
$ALMA_2(0.95)$	5.72	5.05	4.85			
	1752	2087	2239			
$ALMA_2(0.9)$	5.43	5.06	4.90			Gaussian
	2251	2606	2746			
Uniform	6.73	5.53	5.38	5.48	5.43	
	492	578	603	614	621	
Max	6.08	6.38	5.48	5.38	5.38	Homogeneous
	527	607	639	645	647	Polynomial
Prop	6.63	5.98	5.73	5.58	5.43	
	494	575	600	612	615	
MIRA	4.78	4.68	4.63	4.63	4.58	
	3242	3864	4250	4517	4726	

Table 4: Experimental results for the USPS data-set.The test error (top) and number of support patterns (bottom) for five multiclass online algorithms after $j = 1,\ldots,5$ epochs.

Algorithm	Epochs 1	2	3	4	5	Kernel
Perceptron	7.45	5.13	4.60	4.32	3.95	Gaussian
	4215	5635	6469	7023	7359	
$ALMA_2(0.8)$	4.20	3.55	3.27			Poly-Gaussian
	11258	13003	13673			
Uniform	7.07	5.40	4.90	4.88	4.28	Gaussian
	2202	2754	3057	3293	3432	
Max	7.40	6.08	4.63	4.73	4.73	
	2334	2951	3313	3510	3635	
Prop	8.00	7.03	4.98	4.83	4.45	
	2205	2784	3117	3336	3475	
MIRA	3.68	3.08	2.70	2.50	2.38	
	8184	11964	14929	17453	19701	

Table 5: Experimental results for the Letter data-set. The test error (top) and number of support patterns (bottom) for five multiclass online algorithms after $j = 1, \ldots, 5$ epochs.

References

E. L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

J. K. Anlauf and M. Biehl. The adatron: an adaptive perceptron algorithm. *Europhysics Letters*, 10 (7):687–692, Dec 1989.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, 1984.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

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

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

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

T. Friess, N. Cristianini, and C. Campbell. The kernel-adatron: A fast and simple learning procedure for support vector machines. In *Machine Learning: Proceedings of the Fifteenth International Conference*, 1998.

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

A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

D. P. Helmbold and M. K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50:551–573, 1995.

J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, January 1997.

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

N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

C. Mesterharm. A multi-clss linear learning algorithm related to winnow. In *Advances in Neural Information Processing Systems 13*, 1999.

N. J. Nilsson. *Learning Machines: Foundations of trainable pattern classification systems*. McGraw-Hill, New York, 1965.

J. C. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

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