Online Learning of Complex Prediction Problems Using Simultaneous Projections

Yonatan Amit MITMIT@CS.HUJI.AC.IL
The Hebrew University
Jerusalem, 91904, Israel

Shai Shalev-Shwartz SHAI@TTI-C.ORG
Toyota Technological Institute
Chicago, IL 60637, USA

Yoram Singer SINGER@GOOGLE.COM
Google Inc.
Mountain View, CA 94043, USA

Editor: Manfred K. Warmuth

Abstract

We describe and analyze an algorithmic framework for online classification where each online trial consists of *multiple* prediction tasks that are tied together. We tackle the problem of updating the online predictor by defining a projection problem in which each prediction task corresponds to a single linear constraint. These constraints are tied together through a single slack parameter. We then introduce a general method for approximately solving the problem by projecting *simultaneously* and independently on each constraint which corresponds to a prediction sub-problem, and then averaging the individual solutions. We show that this approach constitutes a feasible, albeit not necessarily optimal, solution of the original projection problem. We derive concrete simultaneous projection schemes and analyze them in the mistake bound model. We demonstrate the power of the proposed algorithm in experiments with synthetic data and with multiclass text categorization tasks.

1. Introduction

We discuss and analyze an algorithmic framework for complex prediction problems in the online learning model. Our construction unifies various complex prediction tasks by considering a setting in which at each trial the learning algorithm should make multiple binary decisions. We present a simultaneous online update rule that utilizes the entire set of binary examples received at each trial while retaining the simplicity of algorithms whose update is based on a single binary example.

Online learning is performed in a sequence of consecutive trials. At the beginning of each trial, the algorithm first receives an instance and is required to make a prediction in some complex domain. The prediction is generated using an hypothesis constructed by the algorithm. Once the algorithm makes a prediction it receives the correct target and is allowed to update its hypothesis. In this paper we consider an online learning model in which the complex prediction task can be cast as multiple binary decisions. Such a view is common in Multiclass categorization tasks, for example in Crammer et al. (2006), Crammer and Singer (2003), Allwein et al. (2000). There, the complex problem of predicting which label out of k possible labels is the correct label is cast as a set of binary prediction problems, each of which focuses on two labels.

Previous approaches to this construction can be roughly divided into two paradigms. The first paradigm, which we term the *max update*, tackles the problem by selecting a *single* binary problem and updating the algorithm's hypothesis based on that problem solely. While this approach is sub-optimal, it is often very simple to implement and quite effective in practice. The second approach considers all the binary problems and incorporates the entire information contained for updating the hypothesis. The second approach thus performs an optimal update at the price of often incurring higher computational costs.

We introduce a third approach, which enjoys the simplicity and performance of the max-update approach, while incorporating information expressed in *all* binary problems. Our family of algorithms achieves this goal by considering each instance *separately* and acting *simultaneously*. An update is constructed for each binary sub-problem, and then all the updates are combine together to form the new online hypothesis. As we show in the sequel, the update rule due to each binary example amounts to a projection operation. We thus denote our approach as the *simultaneous projections* approach.

We propose a simple, general, and efficient framework for online learning of a wide variety of complex problems. We do so by casting the online update task as an optimization problem in which the newly devised hypothesis is required to be close to the current hypothesis while attaining a small loss on *multiple* binary prediction problems. Casting the online learning task as a sequence of instantaneous optimization problems was first suggested and analyzed by Kivinen and Warmuth Kivinen and Warmuth (1997) for binary classification and regression problems. In our optimization-based approach, the complex decision problem is cast as an optimization problem that consists of *multiple* linear constraints each of which represents a single binary example. These constraints are tied through a *single* slack variable whose role is to assess the *overall* prediction quality for the complex problem.

The max-update approach described above selects a single binary example, which translates into a single constraint. Performing the update thus becomes a simple projection task, where an analytical solution can often be easily devised. In contrast, the optimal update seeks the optimal solution of the instantaneous optimization problem. However, in the general case no analytical solution can be found, and the algorithm is required to resort to a full scale numeric solver.

We describe and analyze a family of two-phase algorithms. In the first phase, the algorithms solve *simultaneously* multiple sub-problems. Each sub-problem distills to an optimization problem with a *single* linear constraint from the original multiple-constraints problem. The simple structure of each single-constraint problem results in an analytical solution, which is efficiently computable. In the second phase, the algorithms take a convex combination of the independent solutions to obtain a solution for the multiple-constraints problem. We further explore the structure of our problem and attain an update form that combines the two phases while maintaining the simplicity of the simultaneous projection scheme. The end result is an approach whose time complexity and mistake bounds are equivalent to approaches which solely deal with the worst-violating constraint Crammer et al. (2006). In practice, though, the performance of the simultaneous projection framework is much better than update schemes that are based on a single-constraint .

We introduce an additive and multiplicative variants of our framework. The additive framework extends additive algorithms such as the Perceptron Rosenblatt (1958) and the family of Passive-Aggressive algorithms Crammer et al. (2006) to our settings. We then present a multiplicative family of simultaneous algorithms that extends the Winnow family of algorithms Littlestone (1988). We further extend our model showing its applicability when working with a larger family of loss functions. Finally we present a unified analysis in the mistake bound model, based on the primal-dual analysis presented in Shalev-Shwartz and Singer (2006a). Our results are on par with the best known mistake bounds for multiclass algorithms.

Related Work The task of multiclass categorization can be thought of as a specific case of our construction. In multiclass categorization the task is to predict a *single* label out of k possible outcomes. Our simultaneous projection approach is based on the fact that we can retrospectively (after receiving the correct label) cast the problem as the task of making $k - 1$ binary decisions, each of which involves the correct label and one of the competing labels. Our framework then performs an update on each of the problems separately and then combines the updates to form a new hypothesis. The performance of the $k - 1$ predictions is measured through a single loss function. Our approach stands in contrast to previously studied methods which can be roughly be partitioned into three paradigms. The first paradigm follows the max update paradigm presented above. For example, the algorithms by Crammer et al Crammer and Singer (2003), Crammer et al. (2006) focus on the single, worst performing, derived sub-problem. While this approach adheres with the original structure of the problem, the resulting update mechanism is by construction sub-optimal as it oversees almost all of the constraints imposed by the complex prediction problem. (See also Shalev-Shwartz and Singer (2006a) for analysis and explanation of the sub-optimality of this approach.)

Since applying full scale numeric solvers in each online trial is usually prohibitive due to the high computational cost, the optimal paradigm for dealing with complex problems is to tailor a specific efficient solution for the problem on hand. While this approach yielded highly efficient learning algorithms for multiclass categorization problems Crammer and Singer (2003), Shalev-Shwartz and Singer (2006b) and aesthetic solutions for structured output problems Taskar et al. (2003), Tsochantaridis et al. (2004), devising these algorithms required dedicated efforts. Moreover, tailored solutions typically impose rather restrictive assumptions on the representation of the data in order to yield efficient algorithmic solutions.

The third (and probably the simplest) previously studied approach is to break the problem into multiple *decoupled* problems that are solved *independently*. Such translation effectively changes the problem definition. Thus, the simplicity of this approach also underscores its deficiency as it is detached from the original loss of the complex decision problem. Such an approach was used for instance for batch learning of multiclass support vector machines Weston and Watkins (1999) and boosting algorithms Schapire and Singer (1999). Decoupling approaches have further been extended to various ways. Hastie and Tibshirani Hastie and Tibshirani (1998) considered construction of a binary problem for each pair of classes. In Dietterich and Bakiri (1995), Allwein et al. (2000), Crammer and Singer (2002) error correcting output codes are applied to solve the multiclass problem as separate binary problems.

It is interesting to note that the methods for performing multiple projections simultaneously have been studied in a different context in the optimization community. Similar ideas which can be broadly characterized as row-action methods date back more than 50 years, see for example Hildreth (1957), Bregman (1967), Pierro and Iusem (1986). These methods are used to find the optimal solution of a convex function subject to a very large number of constraints. The core idea behind row-action methods is to consider and isolate a few number of constraints and repeatedly perform a projection on a small subset (typically a single constrain) until convergence. Iusem and Depierro Pierro and Iusem (1986) introduced the concept of averaging to the general family of Bregmans methods, where the projection step is relaxed and the new solution is the *average* of the previous solution and the result of the projection. In Censor and Zenios (1997) Censor introduces a parallel version of the row-action methods. The parallel algorithms perform the projection step on each constraint separately and update the new solution to the average of all these projections. For further extensive description of row-action methods see Censor and Zenios (1997). Row-actions methods have recently received attention in the learning community, for problems such as finding the optimal solution of SVM. For instance the SMO technique of Platt Platt (1998) can be viewed as a row-action optimization method that manipulates two constraints at a time. In this paper we take a different approach, and *decompose* a single complex constraint into multiple projections problem which are tied together through a single slack variable.

The rest of the paper is organized as follows. We start with a description of the problem setting in Sec. 2. In Sec. 3 we describe two complex decision tasks that can be tackled by our approach. A template algorithm for additive simultaneous projection in an online learning setting with multiple instances is described in Sec. 4. We propose concrete schemes for selecting an update form in Sec. 5 and analyze our algorithms within the mistake bound model in Sec. 6. We extend our algorithm to a large family of losses in Sec. 7 and derive family of multiplicative algorithms in Sec. 8. We demonstrate the merits of our approach in a series of experiments with synthetic and real datasets in Sec. 9 and conclude in Sec. 10.

2. Problem Setting

In this section we introduce the notation used throughout the paper and formally describe our problem setting. We denote vectors by lower case bold face letters (e.g. $\mathbf{x}$ and $\boldsymbol{\omega}$) where the j'th element of $\mathbf{x}$ is denoted by x_j. We denote matrices by upper case bold face letters (e.g. $\mathbf{X}$), where the j'th row of $\mathbf{X}$ is denoted by $\mathbf{x}_j$. The set of integers $\{1, \ldots, k\}$ is denoted by $[k]$. Finally, we use the hinge function $[a]_+ = \max\{0, a\}$.

Online learning is performed in a sequence of trials. At trial t the algorithm receives a matrix $\mathbf{X}^t$ of size $k_t \times n$, where each row of $\mathbf{X}^t$ is an instance, and is required to make a prediction on the label associated with each instance. We denote the vector of predicted labels by $\hat{\mathbf{y}}^t$. We allow $\hat{y}^t_j$ to take any value in $\mathbb{R}$, where the actual label being predicted is $\text{sign}(\hat{y}^t_j)$ and $|\hat{y}^t_j|$ is the confidence in the prediction. After making a prediction $\hat{\mathbf{y}}^t$ the algorithm receives the correct labels $\mathbf{y}^t$ where $y^t_j \in \{-1, 1\}$ for all $j \in [k_t]$. In this paper we assume that the predictions in each trial are formed by calculating the inner product between a weight vector $\boldsymbol{\omega}^t \in \mathbb{R}^n$ with each instance in $\mathbf{X}^t$, thus $\hat{\mathbf{y}}^t = \mathbf{X}^t \boldsymbol{\omega}^t$. Our goal is to *perfectly* predict the entire vector $\mathbf{y}^t$. We thus say that the vector $\mathbf{y}^t$ was *imperfectly* predicted if there exists an outcome j such that $y^t_j \neq \text{sign}(\hat{y}^t_j)$. That is, we suffer a unit loss on trial t if there exists j, such that $\text{sign}(\hat{y}^t_j) \neq y^t_j$. Directly minimizing this combinatorial error is a computationally difficult task. Therefore, we use an adaptation of the *hinge-loss*, defined $\ell\left(\hat{\mathbf{y}}^t, \mathbf{y}^t\right) = \max_j \left[1 - y^t_j \hat{y}^t_j\right]_+$, as a proxy for the combinatorial error. The quantity $y^t_j \hat{y}^t_j$ is often referred to as the (signed) *margin* of the prediction and ties the correctness and the confidence in the prediction. We use $\ell\left(\boldsymbol{\omega}^t; (\mathbf{X}^t, \mathbf{y}^t)\right)$ to denote $\ell\left(\hat{\mathbf{y}}^t, \mathbf{y}^t\right)$ where $\hat{\mathbf{y}}^t = \mathbf{X}^t \boldsymbol{\omega}^t$. We also denote the set of instances whose labels were predicted incorrectly by $\mathcal{M}^t = \{j \,|\, \text{sign}(\hat{y}^t_j) \neq y^t_j\}$, and similarly the set of instances whose hinge-losses are greater than zero by $\Gamma^t = \{j \,|\, [1 - y^t_j \hat{y}^t_j]_+ > 0\}$.

3. Derived Problems

In this section we further explore the motivation for our problem setting by describing two different complex decision tasks and showing how they can be cast as special cases of our setting. We also would like to note that our approach can be employed in other prediction problems (see Sec. 10).

Multilabel Categorization In the multilabel categorization task each instance is associated with a set of relevant labels from the set $[k]$. The multilabel categorization task can be cast as a special case of a ranking task in which the goal is to rank the relevant labels above the irrelevant ones. Many learning algorithms for this task employ class-dependent features (for example, see Schapire and Singer (2000)). For simplicity, assume that each class is associated with n features and denote by $\phi(\mathbf{x}, r)$ the feature vector for class r. We would like to note that features obtained for different classes typically relay different information and are often substantially different.

A categorizer, or label ranker, is based on a weight vector $\boldsymbol{\omega}$. A vector $\boldsymbol{\omega}$ induces a score for each class $\boldsymbol{\omega} \cdot \phi(\mathbf{x}, r)$ which, in turn, defines an ordering of the classes. A learner is required to build a vector $\boldsymbol{\omega}$ that successfully ranks the labels according to their relevance, namely for each pair of classes (r, s) such that r is relevant while s is not, the class r should be ranked higher than the class s. Thus we require that $\boldsymbol{\omega} \cdot \phi(\mathbf{x}, r) > \boldsymbol{\omega} \cdot \phi(\mathbf{x}, s)$ for every such pair (r, s). We say that a label ranking is imperfect if there exists *any* pair (r, s) which violates this requirement. The loss associated with each such violation is $[1 - (\boldsymbol{\omega} \cdot \phi(\mathbf{x}, r) - \boldsymbol{\omega} \cdot \phi(\mathbf{x}, s))]_+$ and the loss of the categorizer is defined as the maximum over the losses induced by the violated pairs. In order to map the problem to our setting, we define a virtual instance for every pair (r, s) such that r is relevant and s is not. The new instance is the n dimensional vector defined by $\phi(\mathbf{x}, r) - \phi(\mathbf{x}, s)$. The label associated with all of the instances is set to 1. It is clear that an imperfect categorizer makes a prediction mistake on at least one of the instances, and that the losses defined by both problems are the same.

Ordinal Regression In the problem of ordinal regression an instance $\mathbf{x}$ is a vector of n features that is associated with a target rank $y \in [k]$. A learning algorithm is required to find a vector $\boldsymbol{\omega}$ and k thresholds $b_1 \leq \cdots \leq b_{k-1} \leq b_k = \infty$. The value of $\boldsymbol{\omega} \cdot \mathbf{x}$ provides a score from which the prediction value can be defined as the smallest index i for which $\boldsymbol{\omega} \cdot \mathbf{x} < b_i$, $\hat{y} = \min \{i | \boldsymbol{\omega} \cdot \mathbf{x} < b_i\}$. In order to obtain a correct prediction, an ordinal regressor is required to ensure that $\boldsymbol{\omega} \cdot \mathbf{x} \geq b_i$ for all $i < y$ and that $\boldsymbol{\omega} \cdot \mathbf{x} < b_i$ for $i \geq y$. It is considered a prediction mistake if any of these constraints is violated. In order to map the ordinal regression task to our setting, we introduce $k - 1$ instances. Each instance is a vector in $\mathbb{R}^{n+k-1}$. The first n entries of the vector are set to be the elements of $\mathbf{x}$, the remaining $k - 1$ entries are set to $-\delta_{i,j}$. That is, the i'th entry in the j'th vector is set to -1 if $i = j$ and to 0 otherwise. The label of the first $y - 1$ instances is 1, while the remaining $k - y$ instances are labeled as -1. Once we learned an expanded vector in $\mathbb{R}^{n+k-1}$, the regressor $\boldsymbol{\omega}$ is obtained by taking the first n components of the expanded vector and the thresholds $b_1, \ldots, b_{k-1}$ are set to be the last $k - 1$ elements. A prediction mistake of any of the instances corresponds to an incorrect rank in the original problem.

4. Simultaneous Projection Algorithms

Recall that on trial t the algorithm receives a matrix, $\mathbf{X}^t$, of k_t instances, and predicts $\hat{\mathbf{y}}^t = \mathbf{X}^t \boldsymbol{\omega}^t$. After performing its prediction, the algorithm receives the corresponding labels $\mathbf{y}^t$. Each instance-label pair casts a constraint on $\boldsymbol{\omega}_t$, $y_j^t \left(\boldsymbol{\omega}^t \cdot \mathbf{x}_j^t\right) \geq 1$. If all the constraints are satisfied by $\boldsymbol{\omega}^t$ then

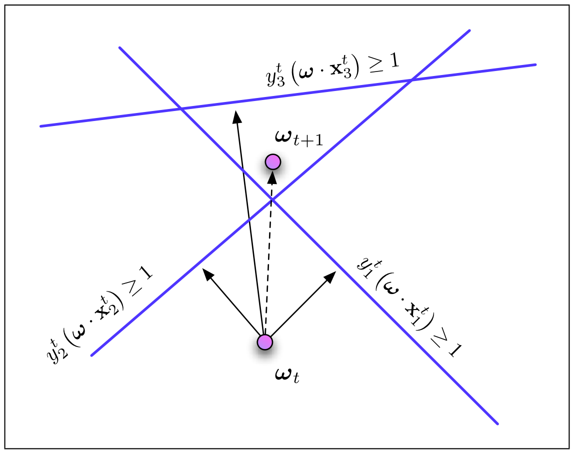

$y_3^t(\boldsymbol{\omega}\cdot\mathbf{x}_3^t)\geq 1$
$\boldsymbol{\omega}_{t+1}$
$y_2^t(\boldsymbol{\omega}\cdot\mathbf{x}_2^t)\geq 1$
$y_1^t(\boldsymbol{\omega}\cdot\mathbf{x}_1^t)\geq 1$
$\boldsymbol{\omega}_t$

Figure 1: Illustration of the simultaneous projections algorithm: each instance casts a constraint on $\boldsymbol{\omega}$ and each such constraint defines a halfspace of feasible solutions. We project on each halfspace in parallel and the new vector is a weighted average of these projections

$\boldsymbol{\omega}^{t+1}$ is set to be $\boldsymbol{\omega}^t$ and the algorithm proceeds to the next trial. Otherwise, we would like to set $\boldsymbol{\omega}^{t+1}$ as close as possible to $\boldsymbol{\omega}^t$ while satisfying all constraints.

Such an aggressive approach may be sensitive to outliers and over-fitting. Thus, we allow some of the constraints to remain violated by introducing a trade-off between the change to $\boldsymbol{\omega}^t$ and the loss attained on $(\mathbf{X}^t, \mathbf{y}^t)$. Formally, we would like to set $\boldsymbol{\omega}^{t+1}$ to be the solution of the following minimization problem,

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^n} \tfrac{1}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega}^t\right\|^2 + C\,\ell\big(\boldsymbol{\omega};\big(\mathbf{X}^t,\mathbf{y}^t\big)\big)\,, \tag{1}$$

where C is a trade-off parameter. As we discuss below, this formalism effectively translates to a cap on the maximal change to $\boldsymbol{\omega}^t$. We rewrite the above optimization by introducing a *single* slack variable as follows:

$$\begin{array}{ll} \displaystyle\min_{\boldsymbol{\omega}\in\mathbb{R}^n,\xi\geq 0} \frac{1}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega}^t\right\|^2 + C\xi & \\ \text{s.t.} \qquad \forall j\in[k_t]: \quad y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\geq 1-\xi \qquad \xi\geq 0 & \end{array} . \tag{2}$$

We denote the objective function of Eq. (2) by $\mathcal{P}^t$ and refer to it as the *instantaneous* primal problem to be solved on trial t. The dual optimization problem of $\mathcal{P}^t$ is the maximization problem

$$\max_{\alpha_1^t,..,\alpha_{k_t}^t} \sum_{j=1}^{k_t}\alpha_j^t - \frac{1}{2}\left\|\boldsymbol{\omega}^t + \sum_{j=1}^{k_t}\alpha_j^t\, y_j^t\,\mathbf{x}_j^t\right\|^2 \quad \text{s.t.} \quad \sum_{j=1}^{k_t}\alpha_j^t \leq C \ \ , \ \ \forall j:\alpha_j^t\geq 0 \quad . \tag{3}$$

The complete derivation is given in Appendix A.

Each dual variable corresponds to a single constraint of the primal problem. The minimizer of the primal problem is calculated from the optimal dual solution as follows, $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t + \sum_{j=1}^{k_t}\alpha_j^t\, y_j^t\,\mathbf{x}_j^t$.

Unfortunately, in the common case, where each $\mathbf{x}_j^t$ is in an arbitrary orientation, there does not exist an analytic solution for the dual problem (Eq. (3)). The difficulty stems from the fact

```
Input:
    Aggressiveness parameter $C > 0$
Initialize:
    $\boldsymbol{\omega}_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots, T$:
    Receive instance matrix $X^t \in \mathbb{R}^{k_t \times n}$
    Predict $\hat{\mathbf{y}}^t = \mathbf{X}^t \boldsymbol{\omega}^t$
    Receive correct labels $\mathbf{y}^t$
    Suffer loss $\ell\left(\boldsymbol{\omega}^t; \left(\mathbf{X}^t, \mathbf{y}^t\right)\right)$
    If $\ell > 0$:
      Choose importance weights $\boldsymbol{\mu}^t$ s.t.
        $\mu_j^t \geq 0$ and $\sum_{j=1}^{k_t} \mu_j^t = 1$
      Choose individual dual solutions $\alpha_j^t$
      Update $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t + \sum_{j=1}^{k_t} \mu_j^t \, \alpha_j^t \, y_j^t \, \mathbf{x}_j^t$
```

Figure 2: Template of simultaneous projections algorithm.

that the sum of the weights α_j^t cannot exceed C. We tackle the problem by breaking it down into k_t reduced problems, each of which focuses on a single dual variable. By doing so we replace the global sum constraint, $\sum_{j=1}^{k_t} \alpha_j^t$, with multiple box constraints, $\alpha_j^t \leq C$, which can easily be dealt with. Formally, for the j'th variable, the j'th reduced problem solves Eq. (3) while fixing $\alpha_{j'}^t = 0$ for all $j' \neq j$. Each reduced optimization problem amounts to the following problem

$$\max_{\alpha_j^t} \; \alpha_j^t - \frac{1}{2} \left\| \boldsymbol{\omega}^t + \alpha_j^t \, y_j^t \, \mathbf{x}_j^t \right\|^2 \quad \text{s.t.} \quad \alpha_j^t \in [0, C] \; . \tag{4}$$

As we demonstrate in the sequel, this reduction into independent problems serves two roles. First, it leads to simple solutions for the reduced problems. Second, and more important, the individual solutions can and are grouped into a feasible solution of the original problem for which we can prove various loss bounds.

We thus next obtain an exact or approximate solution for each reduced problem as if it were independent of the rest. We then choose a non-negative vector $\boldsymbol{\mu} \in \Delta_{k_t}$ where Δ_{k_t} is the k_t dimension probability simplex, formally $\mu_i \geq 0$ and $\sum_{j=1}^{k_t} \mu_j = 1$. Given the vector $\boldsymbol{\mu}$, we multiply each α_j^t by a corresponding value μ_j^t. Our choice of $\boldsymbol{\mu}$ assures us $\{\mu_j^t \alpha_j^t\}$ constitutes a feasible solution to the dual problem defined in Eq. (3) for the following reason. Each $\mu_j^t \alpha_j^t \geq 0$ and the fact that $\alpha_j^t \leq C$ implies that $\sum_{j=1}^{k_t} \mu_j^t \alpha_j^t \leq C$. Finally, the algorithm uses the combined solution and sets $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t + \sum_{j=1}^{k_t} \mu_j^t \, \alpha_j^t \, y_j^t \, \mathbf{x}_j^t$. An illustration of the algorithm is provided in Fig. 4.

5. Solving the reduced problems

We next present four schemes to obtain a solution for the reduced problem (Eq. (4)) and then combine the solution into a single update. The first three schemes provide feasible solutions for the reduced problems and are easy to implement. However, these solutions are not necessarily optimal. In Sec. 5.1 we describe a rather involved yet efficient procedure for finding the optimal

Variant	Choosing μ_j^t	Choosing α_j^t
SimPerc	$\begin{cases} \frac{1}{\lvert\mathcal{M}^t\rvert} & j \in \mathcal{M}^t \\ 0 & \text{otherwise} \end{cases}$	C
ConProj	$\begin{cases} \frac{1}{\lvert\mathcal{M}^t\rvert} & j \in \mathcal{M}^t \\ 0 & \text{otherwise} \end{cases}$	$\min\left\{C, \frac{\ell(\boldsymbol{\omega}^t;(\mathbf{x}_j^t,y_j^t))}{\lVert\mathbf{x}_j^t\rVert^2}\right\}$
ConProj	$\begin{cases} \frac{1}{\lvert\Gamma^t\rvert} & j \in \Gamma^t \\ 0 & \text{otherwise} \end{cases}$	$\min\left\{C, \frac{\ell(\boldsymbol{\omega}^t;(\mathbf{x}_j^t,y_j^t))}{\lVert\mathbf{x}_j^t\rVert^2}\right\}$
SimOpt	See Fig. 4	

Figure 3: Schemes for choosing μ and α.

solution of each reduced problem along side with their weight vector $\boldsymbol{\mu}$ which constitute the means for combining the individual solutions.

Simultaneous Perceptron: The simplest of the update forms generalizes the famous Perceptron algorithm from Rosenblatt (1958) by setting α_j^t to C if the j'th instance is incorrectly labeled, and to 0 otherwise. We then set the weight of μ_j^t to $\frac{1}{|\mathcal{M}^t|}$ for $j \in \mathcal{M}^t$ and to 0 otherwise. We abbreviate this scheme as the *SimPerc* algorithm.

Soft Simultaneous Projections: The soft simultaneous projections scheme uses the fact that each reduced problem has an analytic solution, yielding $\alpha_j^t = \min\left\{C, \ell\left(\boldsymbol{\omega}^t; (\mathbf{x}_j^t, y_j^t)\right) / \left\|\mathbf{x}_j^t\right\|^2\right\}$. We independently assign each α_j^t this optimal solution. We next set μ_j^t to be $\frac{1}{|\Gamma^t|}$ for $j \in \Gamma^t$ and to 0 otherwise. We would like to comment that this solution may update α_j^t also for instances which were correctly classified as long as the margin they attain is not sufficiently large. We abbreviate this scheme as the *SimProj* algorithm.

Conservative Simultaneous Projections: Combining ideas from the above methods, the conservative simultaneous projections scheme optimally sets α_j^t according to the analytic solution. It differs from the SimProj algorithm in the way it selects $\boldsymbol{\mu}^t$. In the conservative scheme only the instances which were incorrectly predicted ($j \in \mathcal{M}^t$) are assigned a positive weight. Put another way, μ_j^t is set to $\frac{1}{|\mathcal{M}^t|}$ for $j \in \mathcal{M}^t$ and to 0 otherwise. We abbreviate this scheme as the *ConProj* algorithm.

5.1 Jointly Optimizing μ and α

Recall that our goal is to propose a feasible solution to Eq. (3) and we do so by independently considering the optimization problem of Eq. (4) for each j. Following, we multiply each α_j^t by a coefficient μ_j^t so that all $\mu_j^t\alpha_j^t$ form a feasible solution. The following analysis shows that the two steps can be unified. For brevity, we omit the superscript t. The task of jointly optimizing both μ and α casts a seemingly non-convex optimization and finding the optimal solution is a priori a hard problem. In this section we derive a somewhat counterintuitive result. By exploring the structure of the problem on hand we show that this joint optimization problem can efficiently be solved in $k_t \log k_t$ time. .

We begin by taking the derivative of the optimal values for α_j while assuming that the values μ_j are fixed and define a convex combination. The reduced problem of Eq. (4) becomes

$$\begin{aligned} \max_{\alpha_j} \ & \mu_j \alpha_j - \frac{1}{2} \left\| \boldsymbol{\omega} + \mu_j \alpha_j \, y_j \, \mathbf{x}_j \right\|^2 \\ \text{s.t.} \quad & \alpha_j \in [0, C] \ , \end{aligned} \tag{5}$$

which can be rewritten as

$$\max_{\alpha_j} \mu_j \alpha_j \left(1 - y_j \left(\boldsymbol{\omega} \cdot \mathbf{x}_j\right)\right) - \frac{1}{2} \mu_j^2 \alpha_j^2 \left\|\mathbf{x}_j\right\|^2 - \frac{1}{2} \left\|\boldsymbol{\omega}\right\|^2 \quad \text{s.t.} \quad \alpha_j \in [0, C] \ .$$

For brevity, we denote the squared norm of $\mathbf{x}_j$ by ν_j. Thus, omitting constants that do not depend on α_j and μ_j, the above optimization problem can be written as

$$\max_{\alpha_j} \ \mu_j \alpha_j \left(1 - y_j \left(\boldsymbol{\omega} \cdot \mathbf{x}_j\right)\right) - \frac{1}{2} \mu_j^2 \alpha_j^2 \nu_j \quad \text{s.t.} \quad 0 \leq \alpha_j \leq C \ . \tag{6}$$

Let us denote the hinge-loss on instance j, $\max\{0, 1 - y_j \left(\boldsymbol{\omega} \cdot \mathbf{x}_j\right)\}$ by ℓ_j. By taking the derivative of the Lagrangian with respect to α_j and equating the result with zero, we get that

$$\alpha_j = \min \left\{ C, \frac{l_j}{\mu_j \nu_j} \right\} .$$

We can now look at two disjoint cases. The first case is when $\alpha_j = \frac{l_j}{\mu_j \nu_j} < C$. In this case α_j takes the value of $\frac{l_j}{\mu_j \nu_j}$ and the value of the optimization problem above becomes

$$\frac{\ell_j^2}{\nu_j} - \frac{1}{2} \frac{\ell_j^2}{\nu_j} = \frac{1}{2} \frac{l_j^2}{\nu_j} .$$

We note in passing that this expression does not depend on μ_j.

The second case is when $\alpha_j = C$. Plugging α_j into Eq. (6) we get the following expression, as a function of μ, which we denote by $f_j(\mu_j)$,

$$f_j(\mu_j) = \mu_j C \ell_j - \frac{1}{2} \mu_j^2 C^2 \nu_j \ . \tag{7}$$

We next take the derivative of f_j above with respect to μ_j and obtain

$$\frac{\partial f}{\partial \mu_j} = C \ell_j - \mu_j C^2 \nu_j \ ,$$

from which we conclude that the optimal value for μ_j is

$$\frac{\ell_j}{C \nu_j} \ .$$

Plugging the optimal value for μ_j back in Eq. (7) we get that the maximum of $f_j(\mu_j)$ is

$$f_j \left(\frac{\ell_j}{C \nu_j} \right) = \frac{\ell_j^2}{\nu_j} - \frac{1}{2} \frac{\ell_j^2}{\nu_j} = \frac{1}{2} \frac{\ell_j^2}{\nu_j} \ .$$

To recap, we may express the optimal value of Eq. (6) as a function of μ_j as follows.

$$f_j(\mu_j) = \begin{cases} \mu_j C\ell_j - \frac{1}{2}\mu_j^2 C^2 \nu_j & \mu_j \le \frac{\ell_j}{C\nu_j} \\ \frac{1}{2}\frac{\ell_j^2}{\nu_j} & \text{otherwise} \end{cases} . \tag{8}$$

Thus, f_j is monotonically increasing in the range $0 \le \mu_j \le \frac{\ell_j}{C\nu_j}$ and is constant for values greater than $\frac{\ell_j}{C\nu_j}$.

Recall that our primary goal is to find a convex combination of μ_j. Thus, we would like to find the optimal assignment to $\boldsymbol{\mu}$ given that $\boldsymbol{\alpha}$ is set optimally. We end up with the following optimization problem.

$$\begin{aligned} &\max \sum_j f_j(\mu_j) \\ &\text{s.t.} \qquad \forall j : \mu_j \ge 0 \qquad \sum_j \mu_j = 1 \end{aligned} . \tag{9}$$

As previously discussed, for all j, f_j increases in the range $0 \le \mu_j \le \frac{\ell_j}{C\nu_j}$ and is constant afterwards. We may use this fact to further classify the structure of the optimal solution of Eq. (9). Assume first that $\sum_j \frac{\ell_j}{C\nu_j} \le 1$. In this case we can increment each, $\mu_j = \frac{\ell_j}{C\nu_j} + B$, where B is a non-negative constant which assures that $\sum_j \mu_j = 1$. This assignment of $\boldsymbol{\mu}$ is clearly optimal, as f_j is increasing and reaches its maximum obtainable value for $\mu_j \ge \frac{\ell_j}{C\nu_j}$.

Now, suppose $\sum_j \frac{\ell_j}{C\nu_j} > 1$. In such a case there exists an optimal solution with $\mu_j \le \frac{\ell_j}{C\nu_j}$ for all j. Suppose in contrary that for every optimal solution $\boldsymbol{\mu}$ there exists some $\hat{j}$ where $\mu_{\hat{j}} = \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}} + \epsilon$ for some non-negative ϵ. Since $\sum_j \frac{\ell_j}{C\nu_j} > 1$ there exists some j' with $\mu_{j'} < \frac{\ell_{j'}}{C\nu_{j'}}$. Since $f_{j'}$ monotonely increases when $\mu_{j'} < \frac{\ell_{j'}}{C\nu_{j'}}$, while $f_{\hat{j}}$ is constant for $\mu_{\hat{j}} > \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}}$ we can create a new assignment $\boldsymbol{\mu}^\star$ increasing the value of the sum $\sum_j f_j(\mu_j)$ by setting $\mu^\star_{\hat{j}} = \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}}$ and add ϵ to $\mu^\star_{j'}$. We thus conclude that for each solution where for some $\mu_{\hat{j}} > \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}}$ there exists an assignment $\boldsymbol{\mu}^\star$ where for all j, $\mu_j \le \frac{\ell_j}{C\nu_j}$ and the objective of Eq. (9) is at least as high.

Thus, when $\sum_j \frac{\ell_j}{C\nu_j} > 1$ we may add the constraint that $\mu_j \le \frac{\ell_j}{C\nu_j}$ and obtain the following optimization problem.

$$\begin{aligned} &\max_\mu \sum_j f_j(\mu_j) \\ &\text{s.t.} \qquad \forall j : 0 \le \mu_j \le \frac{\ell_j}{C\nu_j} \qquad \sum_j \mu_j = 1 \end{aligned}$$

We next introduce non-negative Lagrange multipliers τ, $\{\beta_j\}$, and $\{\eta_j\}$ to obtain the following Lagrangian,

$$L = \sum_j C\ell_j\mu_j - \frac{1}{2}\mu_j^2 C^2 \nu_j - \sum_j \beta_j \mu_j - \tau\left(\sum_j \mu_j - 1\right) + \sum_j \eta_j\left(\mu_j - \frac{\ell_j}{C\nu_j}\right) \tag{10}$$

Taking the derivative with respect to μ_j and comparing to 0 we get the following condition.

$$C\ell_j - \mu_j C^2 \nu_j - \beta_j - \tau + \eta_j = 0,$$

or

$$\mu_j = \frac{C\ell_j - \beta_j - \tau + \eta_j}{C^2 \nu_j} \ .$$

The complementary slackness assures us that when $\mu_j > 0$ then $\beta_j = 0$ and thus

$$\mu_j = \frac{C\ell_j - \tau + \eta_j}{C^2 \nu_j} .$$

Similarly, $\eta_j > 0$ only when $\mu_j = \frac{\ell_j}{C\nu_j}$ and is used only when τ is negative, However, τ may be negative only when $\sum_j \frac{\ell_j}{C\nu_j} < 1$, which we covered before. To summarize, we can write the optimal solution as

$$\mu_j = \max\{0, \frac{C\ell_j - \tau}{C^2 \nu_j}\} \ . \tag{11}$$

We now focus our attention on the task of finding the value of τ. First, note that every value of τ partitions the set $1, \ldots, k_t$ into two sets, indices j whose $\mu_j > 0$ and indices for which $\mu_j = 0$. Formally, let $H = \{j | C\ell_j > \tau\}$ and $L = [k_t] \setminus H$ denote the two sets partitioned according to τ. Clearly $j \in H \iff \mu_j > 0$. Clearly, knowing the value τ allows us to compute the partition to H and L. The converse, however, is also true. Had we known H and L it would have been straightforward to compute τ by using the fact that $\boldsymbol{\mu}$ is in the probability simplex, $\sum_j \mu_j = 1$, to get that

$$\sum_{j \in H} \frac{C\ell_j - \tau}{C^2 \nu_j} = 1 \ . \tag{12}$$

Eq. (12) allows us to easily compute τ and obtain

$$\tau = \frac{\sum_{j \in H} \frac{C\ell_j}{C^2 \nu_j} - 1}{\sum_{j \in H} \frac{1}{C^2 \nu_j}} \ . \tag{13}$$

In order to verify τ serve as a feasible solution, we're required to verify that $\sum_{j \in H} \mu_j = 1$ and that for all $j \in L : \ C\ell_j - \tau \leq 0$. The following lemma states that there is only a single feasible value for τ.

Lemma 1 *Let ℓ_j denote the hinge-loss of instance j, $\max\{0, 1 - y_j (\boldsymbol{\omega} \cdot \mathbf{x}_j)\}$. Let ν_j denote the squared norm of $\mathbf{x}_j$. Denote by f_j the function of μ_j given by $f_j(\mu_j) = \mu_j C\ell_j - \frac{1}{2}\mu_j^2 C^2 \nu_j$. Finally let $\boldsymbol{\mu}$ denote an optimal solution to Eq. (9) computed according to Eq. (11) and $H(\tau)$ denote the set of indices j for which $C\ell_j > \tau$. Then, there is a single value τ that satisfies that $\sum_{j \in H_\tau} \mu_j = 1$ while maintaining that $\forall j \notin H : \mu_j = 0$.*

Proof Suppose by contradiction that there are two feasible values for τ, and denote these values as τ_1 and τ_2. Denote $H(\tau_1)$ and $H(\tau_2)$ by H_1 and H_2 respectively. Assume without loss of generality that $\tau_1 < \tau_2$.

```
Input:
    $\ell_j,\ \nu_j \quad j \in [k_t]$
Algorithm:
    Sort the indices $\{1, \ldots, k_t\}$ by decreasing order of $\ell_j$
    For $i = 2, \ldots, k_t$:
        Define $H = \{1, \ldots, i-1\}$
        Compute $\tau = \frac{\sum_{j \in H} \frac{C\ell_j}{C^2\nu_j} - 1}{\sum_{j \in H} \frac{1}{C^2\nu_j}}$
        Validate $C\ell_i \leq \tau$. If not, continue to next iteration
    Set $\mu_j = \frac{C\ell_j - \tau}{C^2\nu_j}$
    Set $\alpha_j = \min\left\{C, \frac{l_j}{\mu_j\nu_j}\right\}$
```

Figure 4: Calculating $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ efficiently.

First we note that $H_2 \subset H_1$. However,

$$1 = \sum_{j \in H_1} \frac{C\ell_j - \tau_1}{C^2\nu_j} > \sum_{j \in H_1} \frac{C\ell_j - \tau_2}{C^2\nu_j}$$

Where the inequality is due to our assumption that $\tau_1 < \tau_2$. Since $\sum_{j \in H_2} \frac{C\ell_j - \tau_2}{C^2\nu_j}$ must equal 1, we conclude that H_2 must strictly contain more items than H_1. We have thus obtained a contradiction. ■

Using Lemma 1 we can devise an efficient algorithm for finding the optimal value for τ. We first sort the indices $1, \ldots, k_t$ by decreasing order of ℓ_j. Then, for every $i = 2, \ldots, k_t$, we define $H_i = \{1, \ldots, i-1\}$ and compute the value suitable value of τ according to Eq. (13). Finally we verify if $C\ell_i \leq \tau$. The algorithm for finding τ is formally given in Fig. 4.

To recap, we have suggested a mechanism for jointly optimizing both $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$. We showed that it suffices to find a value τ which consistently divides the set $[k_t]$ into two sets as follows. The first set corresponds to indices j for which μ_j is zero and the second includes the non-zero components of $\boldsymbol{\mu}$. Furthermore, we showed that once τ is known, obtaining the vector $\boldsymbol{\mu}$ is a simple task. Last, we described how Lemma 1 translates into an efficient algorithm for finding τ. We thus derived another simultaneous projections scheme, denoted by *SimOpt*, which jointly optimized $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$. This variant of the simultaneous projections framework is guaranteed to yield the largest increase in the dual compared to all other simultaneous projections schemes. We describe empirical results which validate experimentally this property in Sec. 9.

6. Analysis

The algorithms described in the previous section perform updates in order to increase the instantaneous dual problem defined in Eq. (3). We now use the mistake bound model to derive an upper bound on the number of trials on which the predictions of SimPerc and ConProj algorithms are imperfect. Following Shalev-Shwartz and Singer (2006a), the first step in the analysis is to tie the instantaneous dual problems to a global loss function. To do so, we introduce a primal optimization problem defined over the *entire* sequence of examples as follows,

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^n} \frac{1}{2}\left\|\boldsymbol{\omega}\right\|^2 + C\sum_{t=1}^{T} \ell\left(\boldsymbol{\omega};\left(X^t,Y^t\right)\right) \quad .$$

We rewrite the optimization problem as the following equivalent constrained optimization problem,

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^n,\boldsymbol{\xi}\in\mathbb{R}^T} \frac{1}{2}\left\|\boldsymbol{\omega}\right\|^2 + C\sum_{t=1}^{T}\xi_t \quad \text{s.t.} \quad \forall t\in[T], \forall j\in[k_t] : y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\geq 1-\xi_t \quad \forall t:\xi_t\geq 0. \tag{14}$$

We denote the value of the objective function at $(\boldsymbol{\omega},\boldsymbol{\xi})$ for this optimization problem by $\mathcal{P}(\boldsymbol{\omega},\boldsymbol{\xi})$. A competitor who may see the entire sequence of examples in advance may in particular set $(\boldsymbol{\omega},\boldsymbol{\xi})$ to be the minimizer of the problem which we denote by $(\boldsymbol{\omega}^\star,\boldsymbol{\xi}^\star)$. Standard usage of Lagrange multipliers yields that the dual of Eq. (14) is,

$$\max_{\boldsymbol{\lambda}} \sum_{t=1}^{T}\sum_{j=1}^{k_t}\lambda_{t,j} - \frac{1}{2}\left\|\sum_{t=0}^{T}\sum_{j=1}^{k_t}\lambda_{t,j}\, y_j^t\,\mathbf{x}_j^t\right\|^2 \quad \text{s.t.} \quad \forall t:\sum_{j=1}^{k_t}\lambda_{t,j}\leq C \qquad \forall t,j:\lambda_{t,j}\geq 0 \ . \tag{15}$$

We denote the value of the objective function of Eq. (15) by $\mathcal{D}(\boldsymbol{\lambda}_1,\cdots,\boldsymbol{\lambda}_T)$, where each $\boldsymbol{\lambda}_t$ is a vector in $\mathbb{R}^{k_t}$. Through our derivation we use the fact that any set of dual variables $\boldsymbol{\lambda}_1,\cdots,\boldsymbol{\lambda}_T$ defines a feasible solution $\boldsymbol{\omega}=\sum_{t=1}^{T}\sum_{j=1}^{k_t}\lambda_{t,j}y_j^t\mathbf{x}_j^t$ with a corresponding assignment of the slack variables.

Clearly, the optimization problem given by Eq. (15) depends on all the examples from the first trial through time step T and thus can only be solved in hindsight. We note however, that if we ensure that $\lambda_{s,j}=0$ for all $s>t$ then the dual function no longer depends on instances occurring on rounds proceeding round t. As we show next, we use this primal-dual view to derive the skeleton algorithm presented in Fig. 2 by finding a new feasible solution for the dual problem on every trial. Formally, the instantaneous dual problem, given by Eq. (3), is equivalent (after omitting an additive constant) to the following constrained optimization problem,

$$\max_{\boldsymbol{\lambda}}\mathcal{D}(\boldsymbol{\lambda}_1,\cdots,\boldsymbol{\lambda}_{t-1},\boldsymbol{\lambda},\mathbf{0},\cdots,\mathbf{0}) \quad \text{s.t.} \quad \boldsymbol{\lambda}\geq\mathbf{0} \ , \ \sum_{j=1}^{k_t}\lambda_j\leq C \ . \tag{16}$$

That is, the instantaneous dual problem is obtained from $\mathcal{D}(\boldsymbol{\lambda}_1,\cdots,\boldsymbol{\lambda}_T)$ by fixing $\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_{t-1}$ to the values set in previous rounds, forcing $\boldsymbol{\lambda}_{t+1}$ through $\boldsymbol{\lambda}_T$ to the zero vectors, and choosing a feasible vector for $\boldsymbol{\lambda}_t$. Given the set of dual variables $\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_{t-1}$ it is straightforward to show that the prediction vector used on trial t is $\boldsymbol{\omega}^t=\sum_{s=1}^{t-1}\sum_j\lambda_{s,j}y_j^s\mathbf{x}_j^s$. Equipped with these relations and omitting constants which do not depend on $\boldsymbol{\lambda}_t$ Eq. (16) can be rewritten as,

$$\max_{\lambda_1,\ldots,\lambda_{k_t}}\sum_{j=1}^{k_t}\lambda_j-\frac{1}{2}\left\|\boldsymbol{\omega}^t+\sum_{j=1}^{k_t}\lambda_j y_j^t\mathbf{x}_j^t\right\|^2 \quad \text{s.t.} \quad \forall j:\lambda_j\geq 0, \quad \sum_{j=1}^{k_t}\lambda_j\leq C \ . \tag{17}$$

The problems defined by Eq. (17) and Eq. (3) are equivalent. Thus, weighing the variables $\alpha_1^t,\ldots,\alpha_{k_t}^t$ by $\mu_1^t,\ldots,\mu_{k_t}^t$ also yields a feasible solution for the problem defined in Eq. (16), namely $\lambda_{t,j}=\mu_j^t\,\alpha_j^t$. We now tie all of these observations together by using the weak-duality theorem. Our first bound is given for the SimPerc algorithm.

Theorem 2 *Let $\left(\mathbf{X}^1,\mathbf{y}^1\right),\ldots,\left(\mathbf{X}^T,\mathbf{y}^T\right)$ be a sequence of examples where $\mathbf{X}^t$ is a matrix of k_t examples and $\mathbf{y}^t$ are the associated labels. Assume that for all t and j the norm of an instance $\mathbf{x}_j^t$ is at most R. Then, for any $\boldsymbol{\omega}^\star \in \mathbb{R}^n$ the number of trials on which the prediction of SimPerc is imperfect is at most,*

$$\frac{\frac{1}{2}\|\boldsymbol{\omega}^\star\|^2 + C\sum_{t=1}^T \ell\left(\boldsymbol{\omega}^\star;(\mathbf{X}^t,\mathbf{y}^t)\right)}{C-\frac{1}{2}C^2R^2} \ .$$

Proof To prove the theorem we make use of the weak-duality theorem. Recall that any dual feasible solution induces a value for the dual's objective function which is upper bounded by the optimum value of the primal problem, $\mathcal{P}\left(\boldsymbol{\omega}^\star,\boldsymbol{\xi}^\star\right)$. In particular, the solution obtained at the end of trial T is dual feasible, and thus $\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) \le \mathcal{P}(\boldsymbol{\omega}^\star,\boldsymbol{\xi}^\star)$. We now rewrite the left hand-side of the above equation as the following sum,

$$\mathcal{D}(\mathbf{0},\ldots,\mathbf{0}) + \sum_{t=1}^T \left[\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_t,\mathbf{0},\ldots,\mathbf{0}) - \mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_{t-1},\mathbf{0},\ldots,\mathbf{0})\right] \ . \tag{18}$$

Note that $\mathcal{D}(\mathbf{0},\ldots,\mathbf{0})$ equals 0. Therefore, denoting by Δ_t the difference in two consecutive dual objective values, $\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_t,\mathbf{0},\ldots,\mathbf{0})-\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_{t-1},\mathbf{0},\ldots,\mathbf{0})$, we get that $\sum_{t=1}^T \Delta_t \le \mathcal{P}(\boldsymbol{\omega}^\star,\boldsymbol{\xi}^\star)$. We now turn to bounding Δ_t from below. First, note that if the prediction on trial t is perfect ($\mathcal{M}^t = \emptyset$) then SimPerc sets $\boldsymbol{\lambda}_t$ to the zero vector and thus $\Delta_t = 0$. We can thus focus on trials for which the algorithm's prediction is imperfect. We remind the reader that by unraveling the update of $\boldsymbol{\omega}^t$ we get that $\boldsymbol{\omega}^t = \sum_{s<t}\sum_{j=1}^{k_s} \lambda_{s,j} y_j^s \mathbf{x}_j^s$. We now rewrite Δ_t as follows,

$$\Delta_t = \sum_{j=1}^{k_t} \lambda_{t,j} - \frac{1}{2}\left\|\boldsymbol{\omega}^t + \sum_{j=1}^{k_t} \lambda_{t,j} y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2 \ . \tag{19}$$

By construction, $\lambda_{t,j} = \mu_j^t \alpha_j^t$ and $\sum_{j=1}^{k_t} \mu_j^t = 1$, which lets us further expand Eq. (19) and write,

$$\Delta_t = \sum_{j=1}^{k_t} \mu_j^t \alpha_j^t - \frac{1}{2}\left\|\boldsymbol{\omega}^t + \sum_{j=1}^{k_t} \mu_j^t \alpha_j^t y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\sum_{j=1}^{k_t} \mu_j^t \left\|\boldsymbol{\omega}^t\right\|^2 \ .$$

The squared norm, $\|\cdot\|^2$ is a convex function in its vector argument and thus Δ_t is concave, which yields the following lower bound on Δ_t,

$$\Delta_t \ge \sum_{j=1}^{k_t} \mu_j^t \left[\alpha_j^t - \frac{1}{2}\left\|\boldsymbol{\omega}^t + \alpha_j^t y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2\right] \ . \tag{20}$$

The SimPerc algorithm sets μ_j^t to be $1/|\mathcal{M}^t|$ for all $j \in \mathcal{M}^t$ and to be 0 otherwise. Furthermore, for all $j \in \mathcal{M}^t$, α_j^t is set to C. Thus, the right hand-side of Eq. (20) can be further simplified and written as,

$$\Delta_t \ge \sum_{j\in\mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2}\left\|\boldsymbol{\omega}^t + C y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2\right] \ .$$

In order to further explore Eq. (6) we require the following simple lemma

Lemma 3 *Let $\boldsymbol{\omega}^t$ denote the predictor used by the SimPerc scheme on trial t. Let $j \in \mathcal{M}^t$ denote an index of a mispredicted instance on trial t. Then*

$$\frac{1}{2}\left\|\boldsymbol{\omega}^t + C y_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2 \le \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2$$

Proof We start by expanding the norm of the vector after the update,

$$\frac{1}{2}\left\|\boldsymbol{\omega}^t + C y_j^t \mathbf{x}_j^t\right\|^2 = \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2 + C y_j^t \boldsymbol{\omega}^t \cdot \mathbf{x}_j^t + \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 .$$

Thus, the change in the norm is,

$$\begin{aligned}\frac{1}{2}\left\|\boldsymbol{\omega}^t + C y_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2 &= \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2 + C y_j^t \boldsymbol{\omega}^t \cdot \mathbf{x}_j^t + \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2 \\ &= C y_j^t \boldsymbol{\omega}^t \cdot \mathbf{x}_j^t + \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 \ .\end{aligned}$$

The set $\mathcal{M}^t$ consists of indices of instances which were incorrectly classified. Thus, $y_j^t(\boldsymbol{\omega}^t \cdot \mathbf{x}_j^t) \le 0$ for every $j \in \mathcal{M}^t$. The equation above can thus be further bounded by $\frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2$ ■

Lemma 3 assures us that for all instances whose $\mu_j > 0$ the term $\frac{1}{2}\left\|\boldsymbol{\omega}^t + C y_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\boldsymbol{\omega}^t\right\|^2$ can be upper bounded. Therefore, Δ_t can further be bounded from below as follows,

$$\Delta_t \ge \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2\right] \ge \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2}C^2 R^2\right] = C - \frac{1}{2}C^2R^2 \ , \tag{21}$$

where for the second inequality we used the fact that the norm of all the instances is bounded by R. To recap, we have shown that on trials for which the prediction is imperfect $\Delta_t \ge C - \frac{1}{2}C^2R^2$, while in perfect trials where no mistake is made $\Delta_t = 0$. Putting all the inequalities together we obtain the following bound,

$$\left(C - \frac{1}{2}C^2R^2\right)\epsilon \le \sum_{t=1}^{T}\Delta_t = \mathcal{D}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T) \le \mathcal{P}(\boldsymbol{\omega}^\star, \boldsymbol{\xi}^\star) \ , \tag{22}$$

where ϵ is the number of imperfect trials. Finally, rewriting $\mathcal{P}(\boldsymbol{\omega}^\star, \boldsymbol{\xi}^\star)$ as $\frac{1}{2}\|\boldsymbol{\omega}^\star\|^2 + C\sum_{t=1}^{T}\ell(\boldsymbol{\omega}^\star; (\mathbf{X}^t, \mathbf{y}^t)$ yields the bound stated in the theorem. ■

The ConProj algorithm updates the same set of dual variables as the SimPerc algorithm. However, it selects α_j^t to be the optimal solution of Eq. (4). Thus, the value of Δ_t attained by the ConProj algorithm is never lower than the value attained by the SimPerc algorithm, assuming both versions start with the same predictor $\boldsymbol{\omega}_t$. The case of the SimOpt algorithm is very similar, as it promises to optimally increase the value of Δ_t and thus is never lower than the value attained by the SimPerc algorithm. The following corollary is a direct consequence of these observations.

Corollary 4 *Under the same conditions of Thm. 2 and for any $\boldsymbol{\omega}^\star \in \mathbb{R}^n$, the number of trials on which the prediction of either ConProj or SimOpt is imperfect is at most,*

$$\frac{\frac{1}{2}\|\boldsymbol{\omega}^\star\|^2 + C\sum_{t=1}^{T}\ell\left(\boldsymbol{\omega}^\star;(\mathbf{X}^t,\mathbf{y}^t)\right)}{C-\frac{1}{2}C^2R^2} \ .$$

Note that the predictions of the SimPerc algorithm do not depend on the specific value of C, thus for $R=1$ and an optimal choice of C the bound attained in Thm. 2 now becomes.

$$\epsilon \le \ell\left(\boldsymbol{\omega}^\star;(\mathbf{X}^t,\mathbf{y}^t)\right) + \frac{1}{2}\|\boldsymbol{\omega}^\star\|^2 + \frac{1}{2}\sqrt{\|\boldsymbol{\omega}^\star\|^4 + \|\boldsymbol{\omega}^\star\|^2\sum_{t=1}^{T}\ell\left(\boldsymbol{\omega}^\star;(\mathbf{X}^t,\mathbf{y}^t)\right)} \ .$$

See Appendix B for a complete analysis.

We conclude this section with a few closing words about the SimProj variant. The SimPerc and ConProj algorithms ensure a minimal increase in the dual by focusing solely on classification errors and ignoring margin errors. While this approach ensures a sufficient increase of the dual, in practice it appears to be a double edged sword as the SimProj algorithm performs empirically better. This superior empirical performance can be motivated by a viewing the similarity of the update forms performed by the SimProj and SimOpt variants, which means that the actual increase in the dual attained by the SimProj algorithm is larger than can be guaranteed using worst case analysis.

7. Decomposable Losses

Recall that our algorithms tackle complex decision problems by decomposing each instance into multiple binary decision tasks, thus, on trial t the algorithm receives k_t instances. The classification scheme is evaluated by looking at the maximal violation of the margin constraints $\ell\left(\hat{\mathbf{y}}^t,\mathbf{y}^t\right) = \max_j\left[1-y_j^t\hat{y}_j^t\right]_+$. While such approach often captures the inherent relation between the multiple binary tasks, it may often be desired to introduce more complex evaluation measures. In this section we introduce a generalization of the algorithm for various decomposable losses. As a corollary we obtain a *Simultaneous Projection* algorithm that is competitive with the average performance error on each set of k_t instances.

First, we introduce the notion of the decomposable losses. On trial t the algorithms receives a partition of the k_t instances into r_t sets. Let $S_1^t,\ldots,S_{r_t}^t$ denote a partition of $[k_t]$ into r_t sets, namely, $\cup_l S_l^t = [k_t]$ and $\forall l \neq k:\ S_l^t \cap S_k^t = \emptyset$. A set S_l ties the instances in the sense that failing to predict *any* instance in S_l amounts to the same error as failing to predict all of them. We thus suffer a unit loss at trial t for each set S_l that was imperfectly predicted. The definition of the loss is extended to

$$\hat{\ell}\left(\hat{\mathbf{y}}^t,\mathbf{y}^t\right) = \frac{1}{r_t}\sum_{l=1}^{r_t}\max_{j\in S_l^t}\left[1-y_j^t\hat{y}_j^t\right]_+ \ . \tag{23}$$

By construction, the setting suggested in Sec. 2 is a special case of the setting we consider in this section. We show in the sequel though that our original analysis carries over this this more general setting.

Thus, each iteration the algorithm receives k_t instances and a partition of the labels into sets $S_1^t,\ldots,S_{r_t}^t$. The instantaneous primal Eq. (2) is thus extended to include a single slack variable

```
Input:
    Aggressiveness parameter $C > 0$
Initialize:
    $\boldsymbol{\omega}_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots, T$:
    Receive instance matrix $X^t \in \mathbb{R}^{k_t \times n}$
    Predict $\hat{\mathbf{y}}^t = \mathbf{X}^t \boldsymbol{\omega}^t$
    Receive correct labels $\mathbf{y}^t$
    Receive partition of labels $S_1^t, \ldots, S_{r_t}^t$
    Suffer loss $\hat{\ell}\left(\boldsymbol{\omega}^t; \left(\mathbf{X}^t, \mathbf{y}^t\right)\right)$
    If $\hat{\ell} > 0$:
      Choose importance weights $\boldsymbol{\mu}^t$ s.t. for each set $S_l^t$, $\sum_{j \in S_l^t} \mu_j^t = 1$
      Choose individual dual solutions $\alpha_j^t$
      Update $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t + \sum_{l=1}^{r_t} \sum_{j \in S_l^t} \mu_j^t \, \alpha_j^t \, y_j^t \, \mathbf{x}_j^t$
```

Figure 5: The extended simultaneous projections algorithm for decomposable losses.

for *each* set as follows:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\omega} \in \mathbb{R}^n, \xi \geq 0} & \frac{1}{2} \left\|\boldsymbol{\omega} - \boldsymbol{\omega}^t\right\|^2 + \frac{C}{r_t} \sum_{l=1}^{r_t} \xi_l \\ \text{s.t.} & \forall l \in [r_t],\ \forall j \in S_l^t : \quad y_j^t \left(\boldsymbol{\omega} \cdot \mathbf{x}_j^t\right) \geq 1 - \xi_l \qquad \forall l \in [r_t] : \quad \xi_l \geq 0 \end{array} . \tag{24}$$

The dual of Eq. (24) is thus

$$\begin{array}{ll} \max\limits_{\alpha_1^t, .., \alpha_{k_t}^t} & \sum_{j=1}^{k_t} \alpha_j^t - \frac{1}{2} \left\|\boldsymbol{\omega}^t + \sum_{j=1}^{k_t} \alpha_j^t \, y_j^t \, \mathbf{x}_j^t\right\|^2 \\ \text{s.t.} & \forall l : \sum_{j \in S_l^t} \alpha_j^t \leq \frac{C}{r_t} \qquad \forall j : \alpha_j^t \geq 0 \end{array} . \tag{25}$$

Note that since $\forall k \neq l : S_l^t \cap S_k^t = \emptyset$ then the induced constraint $\sum_{j \in S_l^t} \alpha_j^t \leq \frac{C}{r_t}$ corresponds to a unique set of dual variables α_j^t. We can thus apply the same technique and select a non-negative vector $\boldsymbol{\mu}$ where the entries corresponding to each set S_l^t form a probability distribution, namely $\forall l : \sum_{j \in S_l^t} \mu_j^t = 1$. To recap, we can employ any of the variants on each set *separately* and attain a dual feasible solution. We denote these variants as the decomposition variants. In Fig. 5 we provide the pseudo-code of the algorithm.

We next show that our mistake bound analysis can be extended to the decomposable loss. The analysis follows closely to the analysis presented in Sec. 6, where the global primal and global dual are modified so as to use the decomposition loss. We therefore focus only on highlighting the necessary changes. Eq. (14) thus becomes

$$\begin{array}{ll} \min\limits_{\boldsymbol{\omega} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^T} & \frac{1}{2} \left\|\boldsymbol{\omega}\right\|^2 + C \sum_{t=1}^{T} \sum_{l=1}^{r_t} \frac{\xi_{t,l}}{r_t} \\ \text{s.t.} & \forall t \in [T], \forall l \in [r_t], \forall j \in S_l^t : y_j^t \left(\boldsymbol{\omega} \cdot \mathbf{x}_j^t\right) \geq 1 - \xi_{t,l} \qquad \forall t \forall l : \xi_{t,l} \geq 0 \end{array} . \tag{26}$$

and its dual is

$$\max_{\boldsymbol{\lambda}} \sum_{t=1}^{T}\sum_{j=1}^{k_t} \lambda_{t,j} - \frac{1}{2}\left\| \sum_{t=0}^{T}\sum_{j=1}^{k_t} \lambda_{t,j}\, y_j^t\, \mathbf{x}_j^t \right\|^2 \quad \text{s.t.} \quad \forall t \forall l \in [r_t] : \sum_{j\in S_l^t} \lambda_{t,j} \le \frac{C}{r_t} \quad \forall t,j : \lambda_{t,j} \ge 0 \ . \quad (27)$$

Clearly, the instantaneous dual can still be seen as optimizing the global dual, while fixing the dual variables $\lambda_{t',j}$ for all $t' \neq t$.

To recap, we replace the loss of the instantaneous optimization problem defined in Eq. (1) with the average over a decomposition of losses $\hat{\ell}$ as defined by Eq. (23). Next, in order to obtain a mistake bound, we look at the global optimization task defined by Eq. (26). As previously showed, the simultaneous projection scheme can be viewed as an incremental update to the dual of Eq. (26). It is interesting to note that for every decomposition of the k_t instances into sets, the value of $\hat{\ell}\left(\boldsymbol{\omega}; (\mathbf{X}^t, \mathbf{y}^t)\right)$ is upper bounded by $\ell\left(\boldsymbol{\omega}; (\mathbf{X}^t, \mathbf{y}^t)\right)$, as $\hat{\ell}$ is the average over the margin violations while ℓ corresponds to the *worst* margin violation. Thus, the loss underpinning the global optimization from Eq. (14) upper bounds the loss yielding Eq. (26). The following corollary immediately holds.

Corollary 5 *Under the same conditions of Thm. 2, the loss suffered along the run of either decomposition variant is at most,*

$$\frac{\frac{1}{2}\|\boldsymbol{\omega}^\star\|^2 + C\sum_{t=1}^{T} \hat{\ell}\left(\boldsymbol{\omega}^\star; (\mathbf{X}^t, \mathbf{y}^t)\right)}{C - \frac{1}{2}C^2R^2} \ .$$

In conclusion, the simultaneous projection scheme allows us to easily obtain online algorithms and update schemes for complex problems, such algorithms are obtained by decomposing a complex problem into multiple binary problems. It is often the case where the maximal violation over all binary problems correctly captures the inherent violation of the original complex problem. In this section we explored cases where a more refined definition of error is required. Specifically, if we define each binary instance in a separate set, we obtain an algorithmic framework where our competitor is evaluated according to the average loss.

8. Simultaneous Multiplicative Updates

In this section we describe and analyze a multiplicative version of the simultaneous projection scheme. Recall that our motivation was to introduce a solution to the instantaneous optimization problem given in Eq. (2). The instantaneous objective captures the following trade-off. On one hand we would like to set $\boldsymbol{\omega}$ to be as *close* as possible to $\boldsymbol{\omega}^t$. On the other hand, we would like to minimize the loss incurred by the instances received on trial t. In previous sections we used the squared Euclidean norm to define the measure of distance between $\boldsymbol{\omega}^t$ and $\boldsymbol{\omega}$. In this section we take a different approach and use the relative entropy as the notion of closeness between two vectors. By doing so we derive a multiplicative version of our online algorithmic framework. In this section we confine ourselves to linear predictors that lie in the probability simplex, that is, we consider non-negative vectors $\boldsymbol{\omega}$ such that $\sum_{i=1}^{n} \omega_i = 1$. Previously, we used a fixed value of 1 for the margin that is needed in order to suffer no loss, where it was understood that we may simultaneously scale the prediction vector and the margin. Since we now prevent such scaling due to the choice of the simplex domain, we need to slightly modify the definition of the loss and introduce the following definition, $\ell_\gamma\left(\hat{\mathbf{y}}^t, \mathbf{y}^t\right) = \max_j \left[\gamma - y_j^t \hat{y}_j^t\right]_+$.

Recall that on trial t the algorithm receives k_t instances arranged in a matrix $\mathbf{X}^t$. After extending a prediction vector, $\boldsymbol{\omega}^t\mathbf{X}^t$, the algorithm receives the vector of correct labels $\mathbf{y}^t$ and suffers a loss for any incorrect prediction. If no mistake is made the algorithm proceeds to the next round. Otherwise we would like to set $\boldsymbol{\omega}^t$ to be the solution of the following optimization problem

$$\min_{\boldsymbol{\omega}\in\Delta_n} D_{\mathrm{KL}}\left(\boldsymbol{\omega}\|\boldsymbol{\omega}^t\right) + C\,\ell_\gamma\left(\boldsymbol{\omega};\left(\mathbf{X}^t,\mathbf{y}^t\right)\right), \tag{28}$$

where C is a trade-off parameter. The term D_{KL} is the relative entropy operator, also known as the Kullback-Leibler divergence, and is defined as

$$D_{\mathrm{KL}}\left(\boldsymbol{\omega}\|\boldsymbol{\omega}^t\right) = \sum_{i=1}^{n}\boldsymbol{\omega}_i \log\frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t}.$$

The dual problem of Eq. (28) is,

$$\begin{array}{ll} & \gamma\displaystyle\sum_{j=1}^{k_t}\alpha_j^t - \log\left(\sum_{i=1}^{n}\omega_i^t \exp\left(\sum_{j=1}^{k_t}\tau_i^j\right)\right) \\ \text{s.t.} & \displaystyle\sum_{j=1}^{k_t}\alpha_j^t \le C \qquad \forall j:\alpha_j^t\ge 0 \qquad \forall j:\boldsymbol{\tau}^j = \alpha_j^t y_j^t \mathbf{x}_j^t \end{array}. \tag{29}$$

The prediction vector $\boldsymbol{\omega}$ is set as follows,

$$\omega_i = \omega_i^t \frac{\exp\left(\sum_{j=1}^{k_t}\tau_i^j\right)}{\sum_{l=1}^{n}\omega_l^t\exp\left(\sum_{j=1}^{k_t}\tau_i^j\right)}. \tag{30}$$

The complete derivation of the dual problem and the update of $\boldsymbol{\omega}$ is given in Appendix A.

We follow the same technique suggested in Sec. 4 and decompose Eq. (29) into k_t separate problems, each concerning a single dual variable. The resulting j'th reduced dual problem is thus

$$\begin{array}{ll} & \gamma\alpha_j^t - \log\left(\displaystyle\sum_{i=1}^{n}\omega_i^t\exp\left(\tau_i^j\right)\right) \\ \text{s.t.} & 0\le\alpha_j^t\le C \qquad \boldsymbol{\tau}^j = \alpha_j^t y_j^t \mathbf{x}_j^t \end{array}. \tag{31}$$

We next obtain an exact or approximate solution for each reduced problem as if it were independent of the rest. We follow by choosing a vector $\boldsymbol{\mu}\in\Delta_{k_t}$, and multiply each α_j^t by a corresponding value μ_j^t. Our choice of $\boldsymbol{\mu}$ assures us $\{\mu_j^t\alpha_j^t\}$ constitutes a feasible solution to the dual problem defined in Eq. (29) for the following reason. Each $\mu_j^t\alpha_j^t \ge 0$ and the fact that $\alpha_j^t \le C$ implies that $\sum_{j=1}^{k_t}\mu_j^t\alpha_j^t \le C$. Finally, the algorithm uses the combined solution and sets $\boldsymbol{\omega}^{t+1}$ according to Eq. (30). The template of the multiplicative simultaneous projections algorithm is described in Fig. 6.

We may now apply the methods introduced in Sec. 5 and introduce the multiplicative schemes. The SimPerc scheme can be trivially applied to the multiplicative setting. We next show a closed-form solution to α_j^t for each reduced problem if the components of each instance are from $\{-1,0,1\}^n$.

```
Input:
    Aggressiveness parameter $C > 0$
Initialize:
    $\boldsymbol{\omega}_1 = (\frac{1}{n}, \ldots, \frac{1}{n})$
For $t = 1, 2, \ldots, T$:
    Receive instance matrix $X^t \in \mathbb{R}^{k_t \times n}$
    Predict $\hat{\mathbf{y}}^t = \mathbf{X}^t \boldsymbol{\omega}^t$
    Receive correct labels $\mathbf{y}^t$
    Suffer loss $\ell\left(\boldsymbol{\omega}^t; \left(\mathbf{X}^t, \mathbf{y}^t\right)\right)$
    If $\ell > 0$:
      Choose importance weights $\boldsymbol{\mu}^t$ s.t. $\sum_{j=1}^{k_t} \mu_j^t = 1$
      Choose individual dual solutions $\alpha_j^t$
      Compute $\boldsymbol{\tau}^j = \alpha_j^t y_j^t \mathbf{x}_j^t$
      Update $\omega_i^{t+1} = \frac{\omega_i^t \exp\left(\sum_{j=1}^{k_t} \mu_j^t \tau_i^j\right)}{\sum_l \omega_l^t \exp\left(\sum_{j=1}^{k_t} \mu_j^t \tau_l^j\right)}$
```

Figure 6: The multiplicative simultaneous projections algorithm.

To so we need to introduce the following notation.

$$W_j^+ = \frac{\sum_{i: y_j^t x_{ji}^t = 1} \omega_i^t}{\sum_{l=1}^{n} \omega_l^t} \ , \ W_j^- = \frac{\sum_{i: y_j^t x_{ji}^t = -1} \omega_i^t}{\sum_{l=1}^{n} \omega_l^t} \ , \quad \text{and} \ \ W_j^0 = 1 - W_j^+ - W_j^- \quad .$$

The optimal value of α_j^t is thus log of the root of a quadric equation with W_j^+, W_j^-, W_j^0 as coefficients. We also need to take into account the boundary constraints on α_j^t, namely, $0 \le \alpha_j^t \le C$. Thus, α_j^t is the minimum between the following root and C,

$$\log\left(\frac{\gamma W_j^0 + \sqrt{\gamma^2 (W_j^0)^2 + 4(1-\gamma^2) W_j^+ W_j^-}}{2(1-\gamma) W_j^+}\right) \ ,$$

The derivation can be found at Appendix C. Using the closed-form solution for α_j^t we can adapt both the ConProj and SimProj scheme to the multiplicative setting.

We next turn our attention to the analysis of the multiplicative algorithm and focus on the SimPerc scheme. The analysis here follows closely the analysis presented in Sec. 6. Hence, the remainder of this section focuses on highlighting the key changes that are required. Formally, we prove the following theorem.

Theorem 6 *Let* $\left(\mathbf{X}^1, \mathbf{y}^1\right), \ldots, \left(\mathbf{X}^T, \mathbf{y}^T\right)$ *be a sequence of examples where* $\mathbf{X}^t$ *is a matrix of* k_t *examples and* $\mathbf{y}^t$ *are the associated labels. Assume that for all* t *and* j *the* ℓ_∞ *norm of an instance* $\mathbf{x}_j^t$ *is at most* R. *Then, for any* $\boldsymbol{\omega}^\star \in \Delta_n$ *the number of trials on which the prediction of SimPerc is imperfect is at most,*

$$\frac{\sum_{i=1}^n \omega_i^\star \log \frac{\omega_i^\star}{1/n} + C \sum_{t=1}^T \ell_\gamma\left(\boldsymbol{\omega}^\star; \left(\mathbf{X}^t, \mathbf{y}^t\right)\right)}{C - \frac{1}{2} C^2 R^2} \ .$$

Proof Following the technique introduced in Sec. 6, our goal is to upper bound the number of imperfect trials compared to the performance of any fixed competitor, even one defined in hindsight. Our competitor is thus evaluated using the global optimization problem given by,

$$\begin{aligned}
&\min_{\boldsymbol{\omega}\in\Delta_n,\boldsymbol{\xi}\geq 0} \sum_{i=1}^{n} \boldsymbol{\omega}_i \log \frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t} + C\sum_{t=1}^{T} \xi_t \\
&\text{s.t.} \qquad \forall t\in[T], \forall j\in[k_t] : y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right) \geq \gamma - \xi_t \qquad \forall t : \xi_t \geq 0
\end{aligned} \quad . \tag{32}$$

The dual of Eq. (32) is

$$\begin{aligned}
&\gamma \sum_{t=1}^{T}\sum_{j=1}^{k_t} \lambda_j^t - \log\left(\sum_{i=1}^{n} \exp\left(\sum_{t=1}^{T}\sum_{j=1}^{k_t} \tau_i^{tj}\right)\right) \\
&\text{s.t.} \qquad \forall t\in[T] : \sum_{j=1}^{k_t} \lambda_j^t \leq C \qquad \forall t, \forall : \lambda_j^t \geq 0 \qquad \forall t, \forall j : \boldsymbol{\tau}^{tj} = \lambda_j^t y_j^t \mathbf{x}_j^t
\end{aligned} \quad . \tag{33}$$

We denote the objective of Eq. (33) by $\mathcal{D}\left(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T\right)$. The instantaneous dual of Eq. (29) can be seen as incrementally building an assignment for the dual: At trial t we fix $\boldsymbol{\lambda}^s$ for $s<t$ to their previous values, and fix $\boldsymbol{\lambda}^s$ for $s>t$ to 0. Thus $\boldsymbol{\omega}^t$ is defined as follows

$$\omega_i^t = \frac{\exp\left(\sum_{s=1}^{t}\sum_{j=1}^{k_s} \tau_i^{sj}\right)}{\sum_{l=1}^{n} \exp\left(\sum_{s=1}^{T}\sum_{j=1}^{k_s} \tau_l^{sj}\right)} \quad . \tag{34}$$

The key difference between the multiplicative schemes and the previously analyzed scheme lies in Lemma 3. We thus progress to derive a similar lemma for the multiplicative setting.

Lemma 7 *Let $\boldsymbol{\theta} = \sum_{l=1}^{t-1}\sum_{j=1}^{k_l} \lambda_j^t y_j^t \mathbf{x}_j^t$ denote the dual variables assigned in trials prior to t by the SimPerc scheme. Let $j\in\mathcal{M}^t$ denote an index of a mispredicted instance on trial t. Then, the difference,*

$$\log\left(\sum_{i=1}^{n} \exp\left(\theta_i + C x_{ji}^t y_j^t\right)\right) - \log\left(\sum_{i=1}^{n}\exp\left(\theta_i\right)\right) \quad ,$$

is upper bounded by $\frac{1}{2}C^2\left\|\mathbf{x}\right\|_\infty^2$.

Proof Denote the vector $C\mathbf{x}_j^t y_j^t$ by $\boldsymbol{\tau}$. Let $F(\boldsymbol{\theta})$ denote the value of $\log\left(\sum_{i=1}^{n} e^{\theta_i}\right)$. Hence, we would like to upper bound the difference $F(\boldsymbol{\theta}+\boldsymbol{\tau}) - F(\boldsymbol{\theta})$. We prove the lemma based on the following inequality

$$F(\boldsymbol{\theta}+\boldsymbol{\tau}) - F(\boldsymbol{\theta}) \;\leq\; \sum_{i=1}^{n} \frac{e^{\theta_i}}{\sum_l e^{\theta_l}}\tau_i + \frac{1}{2}\max_i \tau_i^2 \quad . \tag{35}$$

The above inequality was utilized and proved in numerous previous analyses of multiplicative update methods. See for instance Examples 2 and 5 in Kivinen and Warmuth (2001). Consider the term $\sum_{i=1}^{n} \frac{e^{\theta_i}}{\sum_l e^{\theta_l}}\tau_i$. Recall that the prediction in trial t is made by using the predictor defined by Eq. (30). Thus, the above term is the following inner product between the vector $\boldsymbol{\tau}$ and the predictor used on round t,

$$\sum_{i=1}^{n} \frac{e^{\theta_i}}{\sum_l e^{\theta_l}}\tau_i = \sum_{i=1}^{n} \omega_i^t \tau_i = \langle \boldsymbol{\omega}^t, \boldsymbol{\tau}\rangle = C y_j^t \langle \boldsymbol{\omega}^t, \mathbf{x}_j^t\rangle \;\leq 0 \;\; ,$$

where the last inequality is due to the fact that we assume $j \in \mathcal{M}^t$ (the prediction was incorrect) and the inner-product is non-positive. Therefore, we obtain the required upper bound

$$F(\boldsymbol{\theta} + \boldsymbol{\tau}) - F(\boldsymbol{\theta}) \leq \frac{1}{2} \max_i \tau_i^2 \frac{1}{2} \left\|\tau\right\|_\infty^2 = \frac{1}{2} C^2 \left\|\mathbf{x}_j^t\right\|_\infty^2 \quad . \tag{36}$$

■

To recap, we showed that the instantaneous dual can be seen as incrementally constructing an assignment for a global dual function (given by Eq. (33). Furthermore, we showed that Lemma 3 can be adapted to the multiplicative settings. The rest of the proof follows the same lines of the proof given in Sec. 6. Namely, trials in which a prediction mistake was made, the SimPerc scheme is guaranteed a substantial increase in the incremental dual buildup. Finally, using weak-duality we obtain that the evaluation measure for the competitor is the lower bounded by,

$$\frac{\sum_{i=1}^n \omega_i^\star \log \frac{\omega_i^\star}{1/n} + C \sum_{t=1}^T \ell_\gamma \left(\boldsymbol{\omega}^\star; (\mathbf{X}^t, \mathbf{y}^t)\right)}{C - \frac{1}{2} C^2 R^2} \quad .$$

■

The multiplicative ConProj scheme assigns α_j^t the value which maximizes the reduced instantaneous dual. The ConProj scheme thus maximizes the difference between the value of the global dual in two consecutive rounds. We thus obtain an equivalent corollary of Corollary 4 for the multiplicative setting.

Corollary 8 *Under the same conditions of Thm. 6 and for any $\boldsymbol{\omega}^\star \in \mathbb{R}^n$, the number of trials on which the prediction of the ConProj scheme is imperfect is at most,*

$$\frac{\sum_{i=1}^n \omega_i^\star \log \frac{\omega_i^\star}{1/n} + C \sum_{t=1}^T \ell_\gamma \left(\boldsymbol{\omega}^\star; (\mathbf{X}^t, \mathbf{y}^t)\right)}{C - \frac{1}{2} C^2 R^2} \quad .$$

We thus showed that the multiplicative SimPerc and ConProj schemes entertain a similar mistake bound as the original formulation. Note, however, that in the multiplicative settings we assume that the ℓ_∞ norm of all instances are bounded by R, while in the additive settings, we have assumed that the ℓ_2 norm of the instances is bounded by R.

9. Experiments

In this section we describe experiments we performed with synthetic and real datasets. The goal of the experiments is to underscore the properties of the simultaneous projection algorithms and to demonstrate some of their merits. Specifically, we examine how the various simultaneous projections variants perform with respect to the size of each block, how does the performance deteriorate with label noise, and the dependency of the algorithms on the number of relevant features. Our experiments with synthetic data are followed with email categorization experiments. On the synthetic data we compare our simultaneous projections algorithms with a commonly used technique whose updates are based on the most violating constraint on each online round (see for instance Crammer et al. (2006)). In the multiclass email categorization experiment, we also use the Sopopo algorithm described in Shalev-Shwartz and Singer (2006b) and a numerical solver which finds the optimal solution of the optimization problem on hand. To recap, we experimented with the following three families of online algorithms.

Simultaneous Projections Algorithms We evaluated all the variants given in Fig. 3, in both their additive and multiplicative forms. We denote the different variants using the notation `name`.A or `name.M` where `name` denotes the specific simultaneous projection scheme as given in Fig. 3 and the .A or .M suffix designate whether the update took an additive or multiplicative form. For example the additive SimProj algorithm is denoted by SimProj.A

MaxPA The algorithm extends the binary Passive-Aggressive family of algorithms Crammer et al. (2006) to structured prediction problems. The algorithm uses a construction which is similar to our instantaneous primal objective Eq. (2), and analogously attempts to obtain a feasible solution to its dual. The difference lies in the fact that the MaxPA algorithm focuses on a single instance whose margin constraint is mostly violated and updates *only* its corresponding dual variable, while fixing all other dual variables to zero. The single dual variable is then optimally computed. This update form constitutes a feasible solution to the instantaneous dual and casts a simple update for the online algorithm.

Optimal Solver The optimal solver algorithm employs a numerical solver on each iteration, and solves optimally the instantaneous primal given by Eq. (2). Specifically, we used the Pegasos algorithm from Shalev-Shwartz et al. (2007) to perform the optimization task. We chose this algorithm since it provides a simple solver which proved superior to second order methods in various classification tasks Shalev-Shwartz et al. (2007).

Sopopo The Sopopo[1] algorithm is a novel algorithm for label ranking and is thus used only in our label ranking experiments with email data. The Sopopo algorithm computes the *optimal* solution to an instantaneous optimization problem similar cast by Eq. (2) while using a slightly different setting. We further elaborate on the different setting in Sec. 9.2.

9.1 Experiments with synthetic data

We tested the performance of the additive and the multiplicative variants of our algorithm in a series of experiments using synthetic data. Our goal in this section is to underscore the merits of our simultaneous projections approach in comparison with the commonly used max update techniques (MaxPA). Since it is often computationally prohibitive to run a full numerical solver on each iteration, we omitted the optimal solver from this set comparisons.

Before we describe the results of our experiments with synthetic data, let us first discuss the procedures used to generate the data. In order to compare both the additive and the multiplicative versions of our framework, we confined ourselves to the more restrictive setting of the multiplicative schemes as described in Sec. 8. Specifically, the data was generated by randomly constructing instances $\mathbf{x}^t \in \{-1, 0, 1\}^n$ and classification was performed by selecting a probability vector $\boldsymbol{\omega} \in \Delta_n$. We used a sparse classifier where the number of relevant features in $\boldsymbol{\omega}$ varied from 10% to 50% active features. The non-zero components of $\boldsymbol{\omega}$ were chosen uniformly at random from $[0, 1]$. We then normalized the vector such that its L_1 norm would be one. We generated linearly separable

1. The name Sopopo stands for SOft Projection Onto Polyhedra.

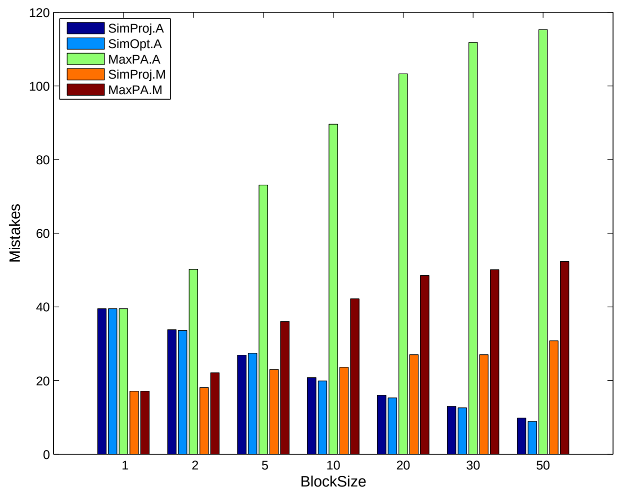


Figure 7: The number of mistakes suffered by the various the additive and multiplicative simultaneous projections methods. The performance of the algorithms is compared as a function of the block size.

data whose margin was calculated as follows. Each entry in $\mathbf{x}$ was set to 0 with probability p and otherwise it was chosen from $\{-1, 1\}$ with equal probability. We then computed the value of γ for which 75% of all instances sampled from the process above would fall inside a margin of γ. We then repeatedly generated and rejected instances, until we managed to construct sufficient number of examples. We refer to a set examples grouped together into a single classification task as a block.

We ran each online experiment for 1000 trials and recorded the number of mistakes performed by the online algorithms. Each experiment was repeated 10 times. The results we present are the averages of these runs. For each experiment, we performed a search for a good value of C. We checked 9 values for C, ranging from 2^{-5} to 2^3. For the multiplicative variants, we also performed a search for a good value of γ by examining values in the range 0.5 to 2 times the margin used during the data generation process. We compared all simultaneous projection variants presented earlier, as well as the multiplicative and additive versions of the MaxPA update.

The first experiment with synthetic data assesses the performance of the various update schemes as a function of the block sizes. We used instances in $\{-1, 0, 1\}^{500}$ where $\boldsymbol{\omega}$ contained 50 relevant features. The results are described in Fig. 7. We clearly see that while both schemes entertain the same mistake bound, in practice the SimProj algorithms always perform better than the maximal update. The difference in practical performance can be attributed to the fact that the simultaneous projections mechanism utilizes more information regarding the structure of the problem at hand.

Note that for both the *MaxPA.A* scheme and the multiplicative schemes the performance deteriorates as the block size increases. A converse trend is exhibited in the case of *SimProj.A* and *SimOpt.A*. One possible explanation for the improvement with block size increase may be observed by considering the geometrical structure of the instances. Recall that we generate uniformly selected linearly separable data. Thus, the update form the additive variants apply can be seen as performing the update using the average instance. For large blocks the average instance is equiv-

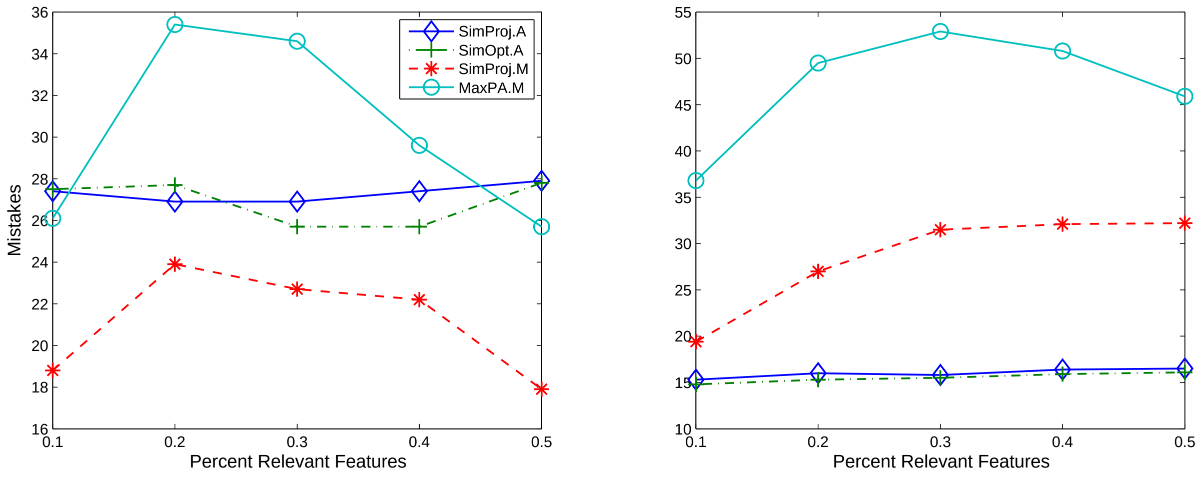
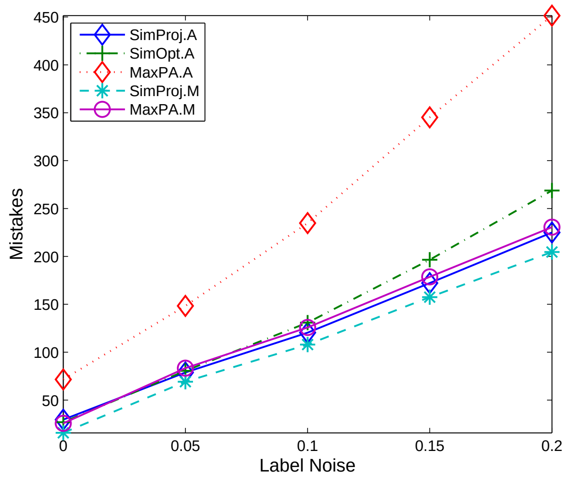


Figure 8: The performance of the additive and multiplicative simultaneous projections algorithms as a function of the sparsity of the hypothesis generating the data. Results are shown for block size of 5 instances (left) and of 20 instances (right).

alent to the vector used to describe the separating hyperplane. Such averaging does not occur in the multiplicative case, or the $MaxPA.A$ scheme.

Our second experiment examines the effect of the sparsity of $\boldsymbol{\omega}$ on the performance of the algorithms. As before, we used instances in $\{-1, 0, 1\}^{500}$. We varied the number of non-zero elements of $\boldsymbol{\omega}$ from 50 to 250. The results of this experiments are plotted in Fig. 8. We ran this experiment with a block size of 5 instances per trial (Fig. 8 left) and a block size of 20 (Fig. 8 right). We omit the plot of $MaxPA.A$ as its performance is much worse than any of the other algorithms. It is apparent that the additive versions are rather insensitive to the sparsity of the prediction matrix. The converse is true for the multiplicative variants. For both block sizes, we see that the performance of the multiplicative versions deteriorate as we increase the percentage of relevant features from 10% to 30%. This decrease in performance is then replaced with a gradual increase in performance once the number of relevant features is over 30%. This increase in performance may be attributed to the fact that there are more relevant features which are set approximately uniformly. Thus, the optimal solution is rather close to the initial vector and the multiplicative algorithm converges faster.

Our last experiment with synthetic data examines the effect of label noise on the performance of the simultaneous projections algorithms. We employed the same settings as in the previous experiment with instances of 500 dimensions and 50 non-zero entries in $\boldsymbol{\omega}$. After the data was generated, we chose to contaminate with label noise each trial with a fixed probability. If the block was selected for contamination, we flipped the label of each the instances in the block with probability 0.4. We repeated the experiment with varying probabilities of picking a block for contamination. We tested values from the set $\{0$ (no label noise), 0.05,0.1,0.15, 0.2$\}$. To avoid too aggressive online updates, we increased the range of the search for C to be in $[2^{-9}, 2^{3}]$. The results of this experiment are plotted in Fig. 9. We ran this experiment with a block size of 5 instances per trials (Fig. 9 left) and block size of 20 (Fig. 9 right) instances per trial.



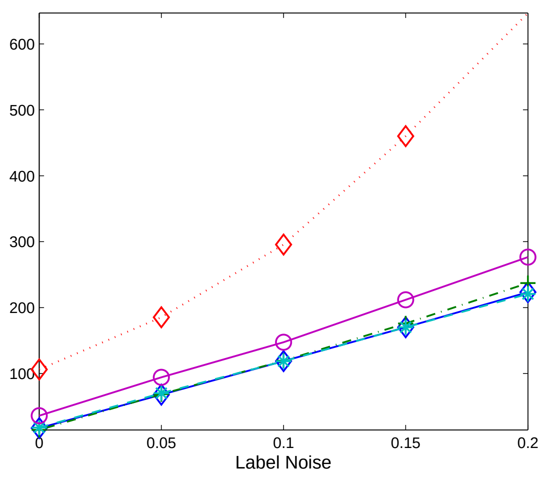


Figure 9: The number of mistakes of the additive and multiplicative simultaneous projections algorithms as a function of the label noise. Results are depicted for block sizes of 5 instances (left) and 20 instances (right) per trial.

username	k	m	Max-SP	SimPerc	ConProj	SimProj	SimOpt	OptSolv
beck-s	101	1972	48.2	48.4	48.8	**43.4**	46.9	45.5
farmer-d	25	3398	25.8	28.8	28.4	25.0	**24.2**	27.5
kaminski-v	41	4478	48.0	47.2	47.6	46.0	44.4	**44.0**
kitchen-l	47	4016	42.5	45.1	43.8	41.4	42.4	**40.3**
lokay-m	11	2490	20.8	24.1	23.9	**18.6**	20.6	20.3
sanders-r	30	1189	18.6	20.9	22.2	**17.7**	19.3	18.2
williams-w3	18	2770	2.8	3.5	3.4	2.7	2.8	**2.6**

Table 1: The percentage of online mistakes of the four additive variants compared to MaxPA and the optimal solver of each instantaneous problem. Experiments were performed on seven users of the Enron data set.

We can clearly see that the number of mistakes of all the SimProj variants scale linearly with the noise rate. It is also apparent that the number of mistakes of the MaxPA.A algorithm scales super linearly with the noise rate. The simultaneous projections variants (both additive and multiplicative) exhibit the best performance. We see that for the smaller block sizes (Fig. 9 left) the best performing version is the *SimProj.M* variant. Note, however, that the variants of SimOpt perform worse than the variants of SimProj . This fact can possibly attributed to the more aggressive update taken by the SimOpt variant when a mistake occurs. As the number of instances per trial increases, the performance of all of simultaneous projections variants is comparable and they all perform better than any of the MaxPA variants.

9.2 Email Classification Experiments

We next tested performance of the different additive and multiplicative simultaneous projection methods described in Sec. 5 on multiclass email categorization tasks and compared them to previously studied algorithms for multiclass categorization. We compared our algorithms to the MaxPA algorithms and to the optimal solver. The experiments were performed with the Enron email dataset. The data set is available from http://www.cs.cmu.edu/∼enron/enron_mail_030204.tar.gz. The learning goal is to correctly classify email messages into user defined folders. Thus, the instances in this dataset are email messages, while the set of classes are the user defined folders denoted by $\{1, \ldots, k\}$. We ran the experiments on the sequence of email messages from 7 different users.

Since each user employs different criteria for email classification, we treated each person as a separate online learning problem. We represented each email message as a vector with a component for every word in the corpus. In order to apply our algorithms, we next describe the class-dependent map we utilize for the additive algorithms. On each trial, and for each class r, we constructed class-dependent vectors as follows. We set $\phi_j(\mathbf{x}^t, r)$ to 2 if the j'th word appeared in the message and it also appeared in a fifth of the messages previously assigned to folder r. Similarly, we set $\phi_j(\mathbf{x}^t, r)$ to -1 if the j'th word appeared in the message but appeared in less than 2 percent of previous messages. In all other cases, we set $\phi_j(\mathbf{x}^t, r)$ to 0. This class-dependent construction is very similar to the construction used in Fink et al. (2006), which yielded high classification accuracy. Next, we employed the mapping described in Sec. 3, and defined a set of $k - 1$ instances for each message as follows. Let the relevant class of an instance be denoted by y. Then, for every irrelevant class $s \neq y$, we define an instance $\mathbf{x}_s^t = \phi(\mathbf{x}^t, y) - \phi(\mathbf{x}^t, s)$ and set its label to 1. All these instances were combined into a single matrix $\mathbf{X}^t$ and were provided to the algorithm in trial t.

For the multiplicative algorithms we took a slightly different approach. Recall that the multiplicative variants assume that the components of each instance are in $\{-1, 0, 1\}$. Hence, in order to adhere to this requirement, $\phi_j(\mathbf{x}^t, r)$ was set to 1 if the j'th word appeared in the message and it also appeared in a fifth of the messages previously assigned to folder r and to 0 in all other cases. Note that this feature generation is performed without knowing the correct label of the instance $\mathbf{x}^t$ and is thus limited to the information available to the online algorithm.

We repeated all tests for 11 values of the trade-off parameter C, testing values from 2^{-5} to 2^5. For each algorithm we present the results for the best choice of C. We first compare the results of the various additive approaches. The results of this experiments are described in Fig. 1. It is apparent that the SimProj and SimOpt variants consistently perform better than the MaxPA variant, and their performance is on par with the performance of the optimal solver. It is interesting to note that in 3 of the 7 users, the SimProj algorithm actually performs better than the optimal solver. The superior performance of the SimProj algorithm may most likely attributed to to the fact that the optimal solver is more aggressive in its update, and is thus more sensitive to noise. We can also see that the SimOpt version, while guaranteeing a larger increase in the global dual, does not necessarily assure better empirical performance. The performances of SimPerc and ConProj are comparable with no obvious winner. Last, we would like to note that the computational cost of the simultaneous projections algorithms is comparable to that of the MaxPA algorithm.

In Fig. 10 we plot the relative number of mistakes of each algorithm with respect to the number of mistakes made by the optimal solver as a function of the number trials for 6 of the 7 users. (The user williams-w3 was omitted as he classifies most his emails into a single folder.) In order to keep the graphs intelligible, we use the optimal solver algorithm as the baseline and plot the difference in performance of the other additive variants. The graphs clearly indicate the superiority of the

SimProj and SimOpt variants over the MaxPA algorithm, and show that SimProj often exhibits the best accuracy.

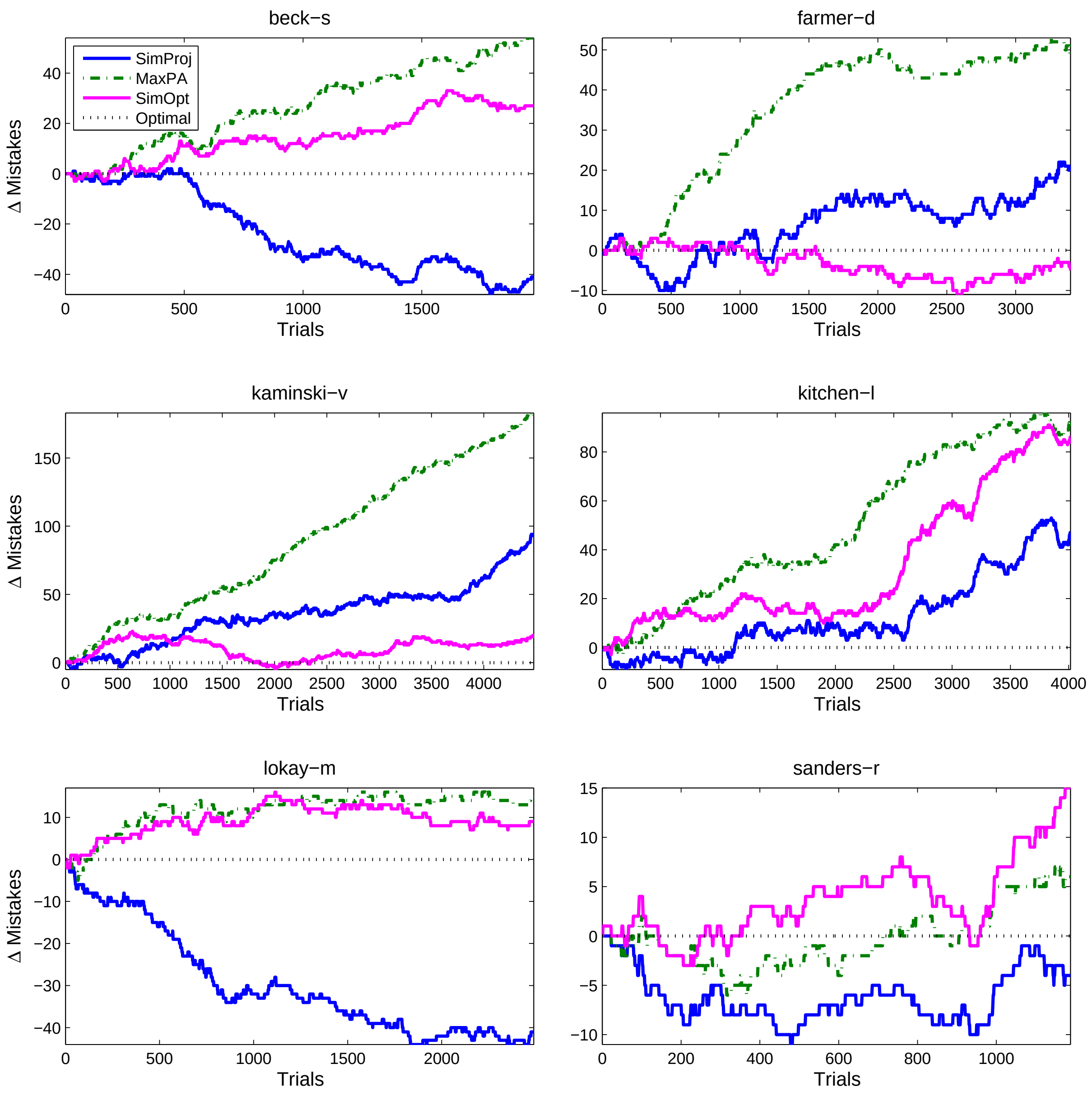


Figure 10: The cumulative number of mistakes of the simultaneous projection algorithms compared with the performance of the Max-PA algorithm and the optimal solver as a function of the number of trials. We plot the difference in the number of mistakes between each algorithm and the optimal solver.

			Additive				Multiplicative			
username	k	m	Max-SP	SimPerc	ConProj	SimProj	Max-SP	SimPerc	ConProj	SimProj
beck-s	101	1972	48.2	48.4	48.8	**43.4**	45.0	43.7	43.6	45.8
farmer-d	25	3398	25.8	28.8	28.4	**25.0**	33.1	35.1	34.8	33.0
kaminski-v	41	4478	48.0	47.2	47.6	**46.0**	50.0	49.8	49.8	49.8
kitchen-l	47	4016	42.5	45.1	43.8	**41.4**	46.8	47.3	47.2	46.5
lokay-m	11	2490	20.8	24.1	23.9	**18.6**	21.8	22.9	22.9	21.4
sanders-r	30	1189	18.6	20.9	22.2	**17.7**	17.8	17.9	17.9	19.3
williams-w3	18	2770	2.8	3.5	3.4	2.7	**2.6**	2.7	**2.6**	2.7

Table 2: The percentage of online mistakes of three additive variants and the MaxPA algorithm compared to their multiplicative counterparts. Experiments were performed on seven users of the Enron data set.

We next compared the performance of the multiplicative and additive variants. The results of this experiments are summarized in Table 2. Observe that the multiplicative versions usually perform on par or slightly worse than the additive versions. This possibly surprising result may be partially attributed to the slightly different feature selection process we used for the multiplicative algorithms. The result also underscores the conjecture that we surfaced above when discussing the synthetic experiments. Namely, the additive simultaneous projections algorithms seem to better capture the structure of the data at hand. The multiplicative versions, however, seem to be less sensitive to the trade-off parameter C taking a preference to the larger values in our test setting.

In our last experiment, we compared the results of our algorithms to the Sopopo algorithm from Shalev-Shwartz and Singer (2006b). The results of this experiment are described in Table 3. Before we discuss the results of this comparison, it is important to note the difference in the model the algorithms use. The algorithms we compare can be roughly divided into two classes of learning algorithms: *single*-prototype algorithms and *multi*-prototype algorithms. As the name imply, the single prototype algorithms maintain a single hypothesis on each online trial. The prediction is obtained by taking the inner-product of the hypothesis with some class dependent map of the instance at hand. The class attaining the highest score is considered the predicted label. All the simultaneous projections algorithms as well as the single prototype version of MaxPA fall into this category. Multi-prototype algorithms take a different approach. On each trial, the online algorithm maintains an hypothesis *for each* possible output class. In order to make a prediction, the algorithm computes the inner product between *each* hypothesis and the instance at hand. The class attaining the largest product is the predicted label. We can see that the various SimProj variants are comparable to the Sopopo algorithm, while the former often performs better (4 of the 7 users we have). It is worth nothing that the Sopopo algorithm exploits the specific settings present in multi-prototype multiclass problems, and efficiently finds the optimum of a projection problem on each trial while our algorithms only find an approximate solution. However, Sopopo is a multi prototype algorithm and thus employs a larger hypothesis space which is more difficult to learn in an online setting. In addition, by employing a single vector representation of the email message, Sopopo cannot benefit from the on-the-fly feature selection which results in class-dependent features.

			Single Prototype					Multi Prototype	
username	k	m	MaxPA	SimPerc	ConProj	SimProj	SimOpt	MaxPA	Sopopo
beck-s	101	1972	48.2	48.4	48.8	**43.4**	46.9	56.0	53.2
farmer-d	25	3398	25.8	28.8	28.4	25.0	24.2	24.0	**23.0**
kaminski-v	41	4478	48.0	47.2	47.6	46.0	44.4	45.9	**43.4**
kitchen-l	47	4016	42.5	45.1	43.8	41.4	42.4	42.2	**40.9**
lokay-m	11	2490	20.8	24.1	23.9	**18.6**	20.6	20.0	19.0
sanders-r	30	1189	18.6	20.9	22.2	**17.7**	19.3	27.9	26.9
williams-w3	18	2770	2.8	3.5	3.4	**2.7**	2.8	4.1	4.1

Table 3: The percentage of online mistakes of four additive simultaneous projection algorithms. The simultaneous projection algorithms are compared with MaxPA (Single-prototype (SP) and Multi-prototype (MP)) and the Sopopo algorithm. Experiments were performed on seven users of the Enron data set.

10. Discussion

We presented a new approach for online categorization with complex output structure. Our algorithms decouples the complex optimization task into multiple sub-tasks, each of which is simple enough to be solved analytically. While the dual representation of the online problem imposes a global constraint on *all* the dual variables, namely $\sum_j \alpha_j^t \leq C$, our framework of simultaneous projections which are followed by averaging the solutions automatically adheres with this constraint and hence constitute a feasible solution. It is worthwhile noting that our approach can also cope with *multiple* constraints of the more general form $\sum_j \nu_j \alpha_j \leq C$, where $\nu_j \geq 0$ for all j. The box constraint implied for each individual projection problem distils to $0 \leq \alpha_j \leq C/\nu_j$ and thus the simultaneous projection algorithm can be used verbatim.

The main scope of this paper is prediction tasks for complex structured decision problems, such as multiclass classification. We approach this problem by first describing the structured problem as a complex optimization problem dealing with multiple binary problems simultaneously. We then use our simultaneous projections scheme to propose a feasible solution to the optimization problem which competes with any decomposition loss (see Sec. 7).

While such an approach is very natural for various structured problems, it is interesting to consider settings in which multiple unrelated binary problems, should be served simultaneously. This approach was the basis for our synthetic experiments, which showed us that the simultaneous projections methods perform much better than the MaxPA approach even though their theoretical bound is similar. One possible explanation of this phenomenon may be attributed to the structure of the space spanned by the examples. In order to illustrate this point, consider for example, the case where all instances on trial t are of similar norm and are orthogonal to each other. The update performed by the simultaneous projections approach is thus optimal. If, on the other hand, all instances received on trial t are exactly the same, then the simultaneous projections approach cannot hope to attain anything better than the MaxPA algorithm. These two extreme cases suggests that further analysis may show that the geometrical structure of the data may shed more light on the progress attained by the simultaneous projections approach.

It is also interesting to explore settings in which the simultaneous projections approach is not immediately applicable. The simultaneous projections approach easily captures the structural requirements expressed by the box constraint $\sum_{j=1}^{k_t} \alpha_j^t \leq C$. While there are many practical problems where such constraints suffice to capture the structure of the problem, more complex constraints are quite prevalent. A notable examples for such a complex setting is the framework of the max-margin Markov (MMM) networks Taskar et al. (2003). While the original learning setting of MMM networks was cast for the batch setting, an equivalent online formulation can be easily obtained. In the MMM framework, the dual problems has dual variables whose number is *exponential* in the original size of the problem. These variables are tied via a simple box constraint. The dual is then transformed into an equivalent form with a much smaller number of variables which are strongly tied with multiple constraints involving all these new variables. While the simultaneous projections approach is well suited for the original formulation, the exponential size of the problem renders such approach unsuitable. On the other hand, the simultaneous projections approach cannot easily construct a feasible dual solution where multiple equality constraints are required. It is thus interesting to explore alternative approaches in which the direct dual whose size is infeasible is reduced to many reduced smaller problems, and only a polynomial subset of which are considered and solved.

References

E. L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47, 2002.

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

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Mar 2006.

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.

M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1): 451–471, 1998.

C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957. Erratum, ibidem, p.361.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Journal of Machine Learning*, 45(3):301–329, July 2001.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

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

A.R. De Pierro and A.N. Iusem. A relaxed version of bregman's method for convex programming. *Journal of Optimization Theory and Applications*, 51:421–440, 1986.

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.

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).).

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):1–40, 1999.

R.E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 32(2/3), 2000.

S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006a.

S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7 (July):1567–1599, 2006b.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.

Appendix A. Derivation of the dual problems

In this section we derive the dual problems of the primal problems presented in the main sections. We start with the derivation of the dual of the optimization problem given in Eq. (2). Using Lagrange multipliers, we rewrite Eq. (2) as follows

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^n,\xi} \max_{\boldsymbol{\alpha}^t\geq 0,\beta\geq 0} \frac{1}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega}^t\right\|^2 + C\xi + \sum_{j=1}^{k_t}\alpha_j^t\left(1-\xi-y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\right) - \beta\xi_t \ .$$

We rearrange the terms in the above equation and rewrite it as follows,

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^n,\xi} \max_{\boldsymbol{\alpha}^t\geq 0,\beta\geq 0} \sum_{j=1}^{k_t}\alpha_j^t + \frac{1}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega}^t\right\|^2 - \sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right) + \xi\left(C-\sum_{j=1}^{k_t}\alpha_j^t-\beta\right) \ . \tag{37}$$

The dual of Eq. (37) is attained by changing the order of the min and max and is given by

$$\max_{\boldsymbol{\alpha}^t\geq 0,\beta\geq 0} \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \sum_{j=1}^{k_t}\alpha_j^t + \frac{1}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega}^t\right\|^2 - \sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right) + \min_{\xi}\xi\left(C-\sum_{j=1}^{k_t}\alpha_j^t-\beta\right) \ . \tag{38}$$

The equation above can be written equivalently as

$$\max_{\boldsymbol{\alpha}^t\geq 0} \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \underbrace{\sum_{j=1}^{k_t}\alpha_j^t + \frac{1}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega}^t\right\|^2 - \sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)}_{\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\omega})} \qquad \text{s.t.} \qquad \sum_{j=1}^{k_t}\alpha_j^t \leq C \ . \tag{39}$$

In order to see that Eq. (39) and Eq. (38) are equivalent, note that the expression

$$\min_{\xi}\xi\left(C-\sum_{j=1}^{k_t}-\beta\right)$$

takes the value of $-\infty$ when $\sum_{j=1}^{k_t}\alpha_j^t+\beta\neq C$. Such an assignment for $\boldsymbol{\alpha}^t$ and β cannot constitute the optimal solution for the maximization problem. The constraint $\beta\geq 0$ thus translates to the constraint $\sum_{j=1}^{k_t}\alpha_j^t\leq C$. Fixing $\boldsymbol{\alpha}^t$, the derivative of $\mathcal{L}$ with respect to $\boldsymbol{\omega}$ is given by

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\omega}} = \boldsymbol{\omega}-\boldsymbol{\omega}^t-\sum_{j=1}^{k_t}\alpha_j^t y_j^t\mathbf{x}_j^t \ .$$

Comparing the derivative to 0, yields the following equation, $\boldsymbol{\omega}=\boldsymbol{\omega}^t+\sum_{j=1}^{k_t}\alpha_j^t y_j^t\mathbf{x}_j^t$. Plugging this equality of $\boldsymbol{\omega}$ Eq. (39) yields the following simplified constrained optimization problem,

$$\begin{aligned}&\max_{\boldsymbol{\alpha}^t\geq 0}\sum_{j=1}^{k_t}\alpha_j^t + \frac{1}{2}\left\|\sum_{j=1}^{k_t}\alpha_j^t y_j^t\mathbf{x}_j^t\right\|^2 - \sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\left(\boldsymbol{\omega}^t+\sum_{l=1}^{k_t}\alpha_l^t y_l^t\mathbf{x}_l^t\right)\cdot\mathbf{x}_j^t\right) \\ &\text{s.t.} \qquad \sum_{j=1}^{k_t}\alpha_j^t\leq C\end{aligned} \ . \tag{40}$$

Rearranging the terms and adding constants which do not depend of $\boldsymbol{\alpha}^t$, we obtain the following dual problem,

$$\begin{aligned} &\max_{\boldsymbol{\alpha}^t} \sum_{j=1}^{k_t} \alpha_j^t - \frac{1}{2}\left\|\boldsymbol{\omega}^t + \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t\right\|^2 \\ &\text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \le C \qquad \forall j \in [k_t] : \ \alpha_j^t \ge 0 \end{aligned} \quad . \tag{41}$$

We now turn our attention to the derivation of the dual of the optimization problem given by Eq. (28). Eq. (28) can be rewritten as follows

$$\begin{aligned} &\min_{\boldsymbol{\omega}\in\Delta_n, \xi\ge 0} \sum_{i=1}^{n} \boldsymbol{\omega}_i \log \frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t} + C\xi \\ &\text{s.t.} \qquad \forall j \in [k_t] : \quad y_j^t \left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right) \ge \gamma - \xi \qquad \xi_l \ge 0 \end{aligned} \quad . \tag{42}$$

We again use Lagrange theory and rewrite the optimization task above as,

$$\min_{\boldsymbol{\omega}\in\Delta_n, \xi\ge 0} \ \max_{\alpha_j^t \ge 0} \sum_{i=1}^{n} \boldsymbol{\omega}_i \log \frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t} + C\xi + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - \xi - y_j^t \left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\right) \quad .$$

Rearranging the terms in the above expression we obtain

$$\min_{\boldsymbol{\omega}\in\Delta_n, \xi\ge 0} \ \max_{\alpha_j^t \ge 0} \sum_{i=1}^{n} \boldsymbol{\omega}_i \log \frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t} + \xi\left(C - \sum_{j=1}^{k_t} \alpha_j^t\right) + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - y_j^t \left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\right) \quad . \tag{43}$$

The dual of Eq. (43) is thus obtained by reversing the order of the min and max and is thus given by

$$\max_{\alpha_j^t \ge 0} \ \min_{\boldsymbol{\omega}\in\Delta_n, \xi\ge 0} \sum_{i=1}^{n} \boldsymbol{\omega}_i \log \frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t} + \xi\left(C - \sum_{j=1}^{k_t} \alpha_j^t\right) + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - y_j^t \left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\right) \quad . \tag{44}$$

The equation above can be rewritten equivalently as follows

$$\begin{aligned} &\max_{\alpha_j^t, \beta^t} \ \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \sum_{i=1}^{n} \boldsymbol{\omega}_i \log \frac{\boldsymbol{\omega}_i}{\boldsymbol{\omega}_i^t} + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - y_j^t \left(\boldsymbol{\omega}\cdot\mathbf{x}_j^t\right)\right) + \beta^t\left(\sum_{i=1}^{n} \omega_i - 1\right) \\ &\text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \le C \qquad \forall j : \alpha_j^t \ge 0 \end{aligned} \quad . \tag{45}$$

In order to see Eq. (45) and Eq. (44) are equivalent, note that the expression $\min_\xi \xi\left(C - \sum_{j=1}^{k_t} \alpha_j^t\right)$ takes the value of $-\infty$ when $\sum_{j=1}^{k_t} \alpha_j^t > C$. Since our goal is to maximize the dual, such solution cannot constitute the optimal assignment for $\boldsymbol{\alpha}^t$. Similarly, when $\sum_{i=1}^{n} \omega_i \ne 1$, the maximization takes the value ∞, thus such a solution cannot constitute the optimal assignment of minimization problem. Finally, we may ignore the constraint $\omega_i \geq 0$ as the optimal solution to the above problem always yields a solution that satisfies this constraint.

In order to further analyze the Eq. (45), let us first denote the vector $\sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t$ by τ. Taking the derivative of Eq. (45) with respect to $\boldsymbol{\omega}$ and comparing the result to 0 yields the following,

$$\omega_i = \omega_i^t e^{\tau_i - 1 + \beta}$$

Recall that β is the Lagrange multiplier associated with the constraint $\sum_{i=1}^n \omega_i = 1$, thus $e^{-1+\beta}$ serves as a normalization constant, and the optimal assignment for ω_i takes the following form.

$$\omega_i = \frac{\omega_i^t e^{\tau_i}}{\sum_{l=1}^n \omega_l^t e^{\tau_l}} . \tag{46}$$

Plugging the value for $\boldsymbol{\omega}$ into Eq. (45) yields the following dual problem for Eq. (42)

$$\max_{\alpha_j^t} \gamma \sum_{j=1}^{k_t} \alpha_j^t - \log\left(\sum_{i=1}^n \omega_i^t e^{\tau_i}\right) \qquad \text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \leq C \qquad \forall j : \alpha_j^t \geq 0 \qquad \tau = \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t \ . \tag{47}$$

Appendix B. Derivation of the SimPerc mistake bound

Thm. 2 assures us that the number of mistakes performed by the SimPerc algorithm is bound by

$$\frac{\frac{1}{2}\|\boldsymbol{\omega}^\star\|^2 + C\sum_{t=1}^T \ell\left(\boldsymbol{\omega}^\star; (\mathbf{X}^t, \mathbf{y}^t)\right)}{C - \frac{1}{2}C^2 R^2} \ . \tag{48}$$

Observe that the prediction made by the SimPerc algorithm does not depend on the value of C. We may thus choose C as to tighten the above bound. Assume $R = 1$ and denote $\sum_{t=1}^T \ell\left(\boldsymbol{\omega}^\star; (\mathbf{X}^t, \mathbf{y}^t)\right)$ by L and $\|\boldsymbol{\omega}^\star\|^2$ by B. It is easy to verify that if $L = 0$, the optimal assignment for C is attained by setting $C = 1$. The bound in this case distills to B. Otherwise, assume $L > 0$. Then, Eq. (48) can be written as

$$\varphi(C) = \frac{\frac{1}{2}B + CL}{C - \frac{1}{2}C^2} \ . \tag{49}$$

The above expression is convex with respect to the parameter C. Hence, in order to find the optimal value of C, it suffices to take the derivative of Eq. (49) with respect to C and compare the result to 0, which yields,

$$\frac{L(C - \frac{1}{2}C^2) - (\frac{1}{2}B + CL)(1 - C)}{\left(C - \frac{1}{2}C^2\right)^2} = 0 \ ,$$

which implies that,

$$LC^2 + BC - B = 0 \tag{50}$$

The largest root of Eq. (50) is given by

$$\begin{aligned} C = \frac{-B + \sqrt{B^2 + 4BL}}{2L} &= \frac{-B + \sqrt{B^2 + 4BL}}{2L} \cdot \frac{-B - \sqrt{B^2 + 4BL}}{-B - \sqrt{B^2 + 4BL}} \\ &= \frac{B^2 - B^2 - 4BL}{2L\left(-B - \sqrt{B^2 + 4BL}\right)} \\ &= \frac{2}{\left(1 + \sqrt{1 + 4\frac{L}{B}}\right)} \end{aligned} \ , \tag{51}$$

It is easy to verify that for $L > 0$ this value of C lies in $(0, 2)$ and thus constitutes the optimal solution of Eq. (49). Plugging Eq. (50) into Eq. (48) yields the following

$$\begin{aligned}
\frac{\frac{1}{2}B + CL}{C - \frac{1}{2}C^2} &= \frac{\frac{B}{C} + L}{1 - \frac{1}{2}C} \\
&= \frac{\frac{1}{2}B\left(1 + \sqrt{1 + 4\frac{L}{B}}\right) + 2L}{2\left(1 - \frac{1}{2}\frac{2}{\left(1+\sqrt{1+4\frac{L}{B}}\right)}\right)} \\
&= \frac{\left(1 + \sqrt{1 + 4\frac{L}{B}}\right)\left(\frac{1}{2}B\left(1 + \sqrt{1 + 4\frac{L}{B}}\right) + 2L\right)}{2\left(\sqrt{1 + 4\frac{L}{B}}\right)} \\
&= \frac{\frac{1}{2}B + 2L}{\frac{1}{2}\left(\sqrt{1 + 4\frac{L}{B}}\right)} + \frac{1}{4}B + \frac{1}{4}B\left(1 + \sqrt{1 + 4\frac{L}{B}}\right) + L \\
&= \frac{1}{2}B + L + \frac{1}{2}\sqrt{B^2 + 4LB}
\end{aligned} \quad . \tag{52}$$

Using the definition of L and B to expand the above expression completes our proof.

Appendix C. An analytic solution for the multiplicative framework

Recall that throughout Sec. 8 section we assumed that the entries of each instance $\mathbf{x}_j^t$ lie in $\{-1, 0, 1\}$. On trial t we would like to find the optimal solution to the reduced problem given by Eq. (31). To do so we recall the notation used in Sec. 8,

$$W_j^+ = \frac{\sum_{i:y_j^t x_{ji}^t = 1} \omega_i^t}{\sum_{l=1}^n \omega_l^t} \ , \ W_j^- = \frac{\sum_{i:y_j^t x_{ji}^t = -1} \omega_i^t}{\sum_{l=1}^n \omega_l^t} \ , \quad \text{and} \ \ W_j^0 = 1 - W_j^+ - W_j^- \quad .$$

The optimization problem defined by Eq. (31) can thus be rewritten as

$$\begin{aligned}
&\gamma\alpha_j^t - \log\left(W_j^+ e^{\alpha_j^t} + W_j^- e^{-\alpha_j^t} + W_j^0\right) \\
&\text{s.t.} \qquad 0 \le \alpha_j^t \le C
\end{aligned} \quad . \tag{53}$$

Taking the derivative of the above equation with respect to α_j^t and comparing the result to zero yields the following,

$$\gamma - \frac{W^+ e^{\alpha_j^t} - W^- e^{-\alpha_j^t}}{W^+ e^{\alpha_j^t} + W^- e^{-\alpha_j^t} + W^0} = 0 \quad . \tag{54}$$

Rearranging terms, Eq. (54) reduces to

$$\gamma\left(W^+ e^{\alpha_j^t} + W^- e^{-\alpha_j^t} + W^0\right) = W^+ e^{\alpha_j^t} - W^- e^{-\alpha_j^t} \ \Rightarrow \ (1-\gamma)W^+ e^{\alpha_j^t} - (1+\gamma)W^- e^{-\alpha_j^t} - \gamma W^0 = 0 \ .$$

For brevity, we denote $e^{\alpha_j^t}$ by β. The equation above is equivalent to the following equation in β,

$$(1-\gamma)W^+\beta - (1+\gamma)W^-\beta^{-1} - \gamma W^0 = 0 \ .$$

Multiplying both sides of the above equation by β, we obtain the following quadratic equation

$$(1-\gamma)W^+\beta^2 - \gamma W^0\beta - (1+\gamma)W^- \ ,$$

whose largest root (the second root is negative) is

$$\beta = \frac{\gamma W^0 + \sqrt{\gamma^2(W^0)^2 + 4(1-\gamma^2)W^+W^-}}{2(1-\gamma)W^+} \ . \tag{55}$$

Since α_j^t must reside in $[0, C]$ we set α_j^t to be the minimum between $\log(\beta)$ and C, yielding,

$$\alpha_j^t = \min\left\{C, \log\left(\frac{\gamma W^0 + \sqrt{\gamma^2(W^0)^2 + 4(1-\gamma^2)W^+W^-}}{2(1-\gamma)W^+}\right)\right\} \ .$$