
Online Classification for Complex Problems Using Simultaneous Projections

Yonatan Amit¹ Shai Shalev-Shwartz¹ Yoram Singer^{1,2}

¹ School of Computer Sci. & Eng., The Hebrew University, Jerusalem 91904, Israel

² Google Inc. 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
{mitmit, shais, singer}@cs.huji.ac.il

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 by defining an instantaneous projection problem in which all the prediction tasks are tied through a single slack parameter. We then introduce a general method for approximately solving the problem by projecting *simultaneously* and independently each constraint which corresponds to a prediction sub-problem, and then average the individual solutions. We show that this approach constitutes a feasible, albeit not necessarily optimal, solution for the original projection problem. We derive concrete simultaneous projection variants and analyze them in the mistake bound model. We demonstrate the power of the proposed algorithm in experiments with online multiclass text categorization. Our experiments indicate that a combination of class-dependent features with the simultaneous projection method outperforms previous algorithms for this task.

1 Introduction and Problem Settings

In most supervised machine learning tasks the goal is to devise an accurate prediction mechanism when the set of possible outcomes is non-binary and often adheres to structural constraints. Notable examples are multiclass categorization, multilabel ranking, and ordinal regression. The common approach for devising learning algorithms for decision problems with complex output structures is to conceptually break the complex problem into simpler sub-problems and then tie the solution through a specially tailored mechanism. See for instance [1, 2, 3, 4] which discuss specific algorithms for multiclass problems and general structured output problems. These and other algorithms are efficient and elegant yet it is not straightforward to extend the algorithms and export them to other complex decision settings. Moreover, the optimization problems imposed by these approaches are rather involved and require either special purpose algorithms (e.g. [2]) or a solver which *sequentially* decomposes the problem in a batch setting [3, 4] and is thus inadequate in online applications. A notable and quite generic example is multiclass categorization with class-dependent features. This construction is used for instance in multiclass versions of AdaBoost [5] where each weak-hypothesis provides a different prediction for each possible outcome (class). In this paper we suggest a new approach that breaks a single complex decision problem into multiple sub-problems which are tied through a *single* slack parameter. While the resulting combined problem is seemingly difficult to solve, we show that by solving the sub-problems independently and *simultaneously* and then averaging the solutions we are able to obtain an approximate feasible solution to the original (complex) task. The combined solution bears formal properties and can be efficiently used in online settings.

We now formally introduce our notation and describe the problem setting. We denote scalars by lower case letters (e.g. x) and vector by bold face letters (e.g. \mathbf{x}). For a vector \mathbf{x} we denote by x_l the l 'th coordinate of \mathbf{x} . Sets are denoted by upper case Latin letters (e.g. A). Elements of a set are indexed by subscripts (e.g. a_s for scalars and \mathbf{a}_s for vectors). For any natural number k , we denote the set $\{1, 2, \dots, k\}$ by $[k]$. Finally, we denote the hinge function by $[a]_+ = \max\{0, a\}$.

Online learning is performed in a sequence of trials. Let $\mathcal{X} \subset \mathbb{R}^n$ denote an instance domain. At trial t the algorithm receives a set of k_t instances $X^t = \{\mathbf{x}_j^t\}_{j=1}^{k_t}$ and is required to make a prediction on the label associated with each instance. We denote the set of predicted labels by $\hat{Y}^t = \{\hat{y}_j^t\}_{j=1}^{k_t}$. We allow \hat{y}_j^t to take any value in \mathbb{R} , where the actual label being predicted is $\text{sign}(\hat{y}_j^t)$ while $|\hat{y}_j^t|$ is the confidence in the prediction. After extending the prediction set \hat{Y}^t the algorithm receives the correct labels $Y^t = \{y_j^t\}_{j=1}^{k_t}$ where $y_j^t \in \{-1, 1\}$ 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 $\omega^t \in \mathbb{R}^n$ with each instance \mathbf{x}_j^t , thus $\hat{Y}^t = \{\omega^t \cdot \mathbf{x}_j^t\}_{j=1}^{k_t}$. Our goal is to *perfectly* predict the entire set Y^t . We thus say that the predicted set \hat{Y}^t is imperfect if there exists even a single outcome j such that $y_j^t \neq \text{sign}(\hat{y}_j^t)$. We evaluate and analyze our algorithms using two evaluation measures. The first indicates whether the prediction is indeed imperfect, namely, we suffer a unit loss on trial t when *any* of the predictions \hat{y}_j^t disagrees with y_j^t . Since minimizing this combinatorial error directly is computationally difficult, as a second evaluation measure we employ an adaptation of the *hinge-loss* to our setting, and define $\ell(\hat{Y}^t, Y^t) = \max_{j \in [k_t]} [1 - y_j^t \hat{y}_j^t]_+$. The quantity $y_j^t \hat{y}_j^t$ is often referred to as the (signed) *margin* of the prediction and binds together the correctness and confidence of the prediction. We use $\ell(\omega; (X^t, Y^t))$ to denote $\ell(\hat{Y}^t, Y^t)$ where $\hat{y}_j^t = \omega \cdot \mathbf{x}_j^t$. We also denote the set of instances whose labels were predicted incorrectly by $\mathcal{M}^t = \{j \mid \text{sign}(\hat{y}_j^t) \neq y_j^t\}$, and similarly the set of instances whose hinge-losses are not zero $\Gamma^t = \{j \mid [1 - y_j^t \hat{y}_j^t]_+ > 0\}$. Finally, we denote the number of trials on which a prediction mistake was made by $\epsilon = |\{t \mid \mathcal{M}^t \neq \emptyset\}|$. Our goal is to construct an algorithm which attains a low value for ϵ relatively to the cumulative loss $\sum_{t=1}^T \ell(\omega; (X^t, Y^t))$ of any competing hypothesis ω , even one which is defined in hindsight.

This paper is organized as follows. In Sec. 2 we start by casting two complex decision tasks to our settings. In Sec. 3 we develop the simultaneous projections algorithm, which we later analyze in Sec. 4. We demonstrate the merits of our approach on a series of experiments in Sec. 5 and conclude in Sec. 6.

2 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 may be beneficial for other prediction problems (see Sec. 6).

Multilabel Categorization In the multilabel categorization task, also referred to as label ranking, each instance is associated with a set of relevant labels from the set $[k]$. Many learning algorithms for this task employ class-dependant features (for example, see [6]). 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 radically different. A categorizer, or label ranker, is based on a weight vector ω . A vector ω casts a score for each class $\omega \cdot \phi(\mathbf{x}, r)$ which, in turn, defines an ordering of the classes. A learner is required to build a vector ω 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 $\omega \cdot \phi(\mathbf{x}, r) > \omega \cdot \phi(\mathbf{x}, s)$ 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 $[1 - (\omega \cdot \phi(\mathbf{x}, r) - \omega \cdot \phi(\mathbf{x}, s))]_+$ and the loss of the categorizer is defined as the maximal loss of any violated pair. 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 corresponds to a prediction mistake on one of the instances, and that the losses defined on both problem are the same.

Ordinal Regression In the problem of ordinal regression each instance, \mathbf{x} , is a vector over n features and is associated with a rank in $y \in [k]$. A learning algorithm is required to find a vector ω and k thresholds $b_1 \leq \dots \leq b_{k-1} \leq b_k = \infty$. The value of $\omega \cdot \mathbf{x}$ provides a score from which the prediction value can be defined as the minimal i for which $\omega \cdot \mathbf{x} < b_i$, namely $\hat{y} = \min \{i \mid \omega \cdot \mathbf{x} < b_i\}$. In order to obtain a correct prediction, a categorizer is required to ensure that

$\omega \cdot \mathbf{x} \geq b_i$ for all $i < y$ and that $\omega \cdot \mathbf{x} < b_i$ 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 settings, we introduce $k - 1$ instances. Each instance is a vector in an \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 $-\delta_{i,j}$, namely 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 set to 1 , while the remaining instances are labeled -1 . Once we obtained an extended vector (in \mathbb{R}^{n+k-1}) using our algorithms, the regressor ω is obtained by taking the first n components of the extended vector and the thresholds b_1, \dots, b_{k-1} are taken from the last $k - 1$ elements. A prediction mistake of any of the instances corresponds to an incorrect rank in the original problem.

3 A Primal-Dual Algorithm

Recall that on trial t the algorithm receives a set of k_t instances, denoted by $X^t = \{\mathbf{x}_j^t\}_{j=1}^{k_t}$ and predicts $\hat{Y}^t = \{\omega^t \cdot \mathbf{x}_j^t\}_{j=1}^{k_t}$. After extending its prediction, the algorithm receives the corresponding labels $Y^t = \{y_j^t\}_{j=1}^{k_t}$. Each such instance-label pair casts a constraint on ω^t , namely $y_j^t (\omega^t \cdot \mathbf{x}_j^t) \geq 1$. If all the constraints are satisfied by ω^t then ω^{t+1} is set to be ω^t and the algorithm proceeds to the next trial. Otherwise, we would like to set ω^{t+1} as close as possible to ω^t while satisfying all constraints.

Such aggressive approach may be sensitive to outliers and over-fitting. Thus, we allow some of the constraints to remain violated, by introducing a *single* slack variable. As we discuss promptly this effectively translates to a cap on the maximal change to ω^t . Formally we would wish to set ω^{t+1} to be the solution of the following optimization problem.

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

We denote the objective function of Eq. (1) 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

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

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

Unfortunately, in the typical case, where the \mathbf{x}_j^t are in an arbitrary orientation, there does not exist an analytic solution for the dual problem (Eq. (2)). We tackle the problem by breaking it down into k_t reduced problems, each of which focuses on a single dual variable. Formally, for the j 'th variable, the j 'th reduced problem solves Eq. (2) while fixing $\alpha_{j'}^t = 0$ for all $j' \neq j$. The optimization problem amounts to the following constrained optimization problem

$$\max_{\alpha_j^t} \alpha_j^t - \frac{1}{2} \left\| \omega^t + \alpha_j^t y_j^t \mathbf{x}_j^t \right\|^2 \quad \text{s.t.} \quad \alpha_j^t \in [0, C]. \quad (3)$$

We now obtain an exact or approximate solution to each reduced problem as if it was independent of the rest. We then choose a distribution $\boldsymbol{\mu}^t \in \Delta_{k_t}$, where $\Delta_{k_t} = \{\boldsymbol{\mu} \in \mathbb{R}^{k_t} : \sum_j \mu_j = 1, \mu_j \geq 0\}$ is the probability simplex, and multiply each α_j^t by the corresponding μ_j^t . This yields a feasible solution to the dual problem defined in Eq. (2) as follows. Each $\mu_j^t \alpha_j^t \geq 0$ and the choice of $\boldsymbol{\mu}^t$ combined with the fact that $0 \leq \alpha_j^t \leq C$ implies that $\sum_{j=1}^{k_t} \mu_j^t \alpha_j^t \leq C$. Finally, the algorithm uses the combined solution and sets $\omega^{t+1} = \omega^t + \sum_{j=1}^{k_t} \mu_j^t \alpha_j^t y_j^t \mathbf{x}_j^t$.

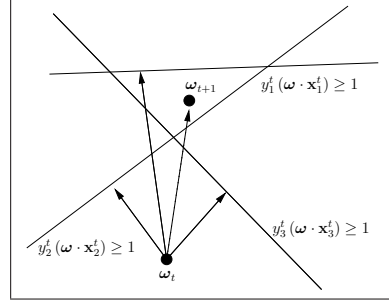


Figure 1: Illustration of the simultaneous projections algorithm: each instance casts a constraint on ω and each such constraint defines a half-space of feasible solutions. We project on each halfspace in parallel and the new vector is a weighted average of these projections

<p>Input: Aggressiveness parameter $C > 0$,</p> <p>Initialize: $\omega_1 = (0, \dots, 0)$</p> <p>For $t = 1, 2, \dots, T$:</p> <p>Receive instance set $X^t = \{\mathbf{x}_j^t\}_{j=1}^{k_t}$</p> <p>Predict $\hat{Y}^t = \{\omega^t \cdot \mathbf{x}_j^t\}_{j=1}^{k_t}$</p> <p>Receive correct labels $Y^t = \{y_j^t\}_{j=1}^{k_t}$</p> <p>Suffer loss $\ell(\omega^t; (X^t, Y^t))$</p> <p>if $\ell > 0$:</p> <p>Choose importance weights $\mu^t \in \Delta_{k_t}$</p> <p>Choose learning parameter α_j^t</p> <p>Update $\omega^{t+1} = \omega^t + \sum_{j=1}^{k_t} \mu_j^t \alpha_j^t y_j^t \mathbf{x}_j^t$</p>
--

Figure 2: Simultaneous projections algorithm.

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 isn't sufficiently large. We abbreviate this variant as the *SimProj* algorithm.

Conservative Simultaneous Projections: Combining ideas from both methods, the conservative simultaneous projections variant optimally sets α_j^t according to the analytic solution. The difference with the *SimProj* algorithm lies in the selection of μ^t . In the conservative simultaneous projections algorithm only the instances which were incorrectly predicted ($j \in \mathcal{M}^t$) are assigned a positive weight. Put differently, μ_j^t is set to $\frac{1}{|\mathcal{M}^t|}$ for $j \in \mathcal{M}^t$ and to 0 otherwise. We abbreviate this variant as the *ConProj* algorithm.

To recap, on each trial t we obtain a feasible solution for the instantaneous dual as described in Eq. (2). This solution combines independently calculated α_j^t , according to a weight vector $\mu^t \in \Delta_{k_t}$. While this solution may not be optimal, it does constitute an infrastructure for obtaining a mistake bound and, as we demonstrate in Sec. 5, performs well in practice.

4 Analysis

The algorithms described in the previous section perform updates so as to increase an instantaneous dual problem as defined in Eq. (2). In this section we show that this approach can be analyzed in the mistake bound model. Specifically, we derive an upper bound on the number of trials on which the predictions of *SimPerc* and *ConProj* algorithms are imperfect.

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, $\min_{\omega \in \mathbb{R}^n} \frac{1}{2} \|\omega\|^2 + C \sum_{t=1}^T \ell(\omega; (X^t, Y^t))$. We rewrite the optimization problem as the following equivalent constrained optimization problem,

$$\min_{\omega \in \mathbb{R}^n, \xi \in \mathbb{R}^T} \|\omega\|^2 + C \sum_{t=1}^T \xi_t \quad \text{s.t.} \quad \forall t \in [T], \forall j \in [k_t] : y_j^t (\omega \cdot \mathbf{x}_j^t) \geq 1 - \xi_t \quad \forall t : \xi_t \geq 0. \quad (4)$$

We denote value of the objective function at (ω, ξ) for this optimization problem by $\mathcal{P}(\omega, \xi)$. An adversary who may see the entire sequence of example in advance may in particular set (ω, ξ) to be the minimizer of the problem which we denote by (ω^*, ξ^*) . Standard usage of Lagrange multipliers yields that the dual of Eq. (4) is,

$$\max_{\lambda} \sum_{t=1}^T \sum_{j=1}^{k_t} \lambda_{t,j} - \frac{1}{2} \left\| \sum_{t=0}^T \sum_{j=1}^{k_t} \lambda_{t,j} y_j^t \mathbf{x}_j^t \right\|^2 \quad \text{s.t.} \quad \forall t : \sum_{j=1}^{k_t} \lambda_{t,j} \leq C \quad \forall t, j : \lambda_{t,j} \geq 0. \quad (5)$$

We denote the value of the objective function of Eq. (5) by $\mathcal{D}(\lambda_1, \dots, \lambda_T)$, where each λ_t is a vector in \mathbb{R}^{k_t} . Through our derivation we use the fact that any set of dual variables $\lambda_1, \dots, \lambda_T$

We next present three variants for obtaining a solution for the reduced problem given in Eq. (3) and combining the solution into a single update.

Simultaneous Perceptron: The simplest of the update forms generalizes the famous Perceptron algorithm from [7] by setting α_j^t to C if the j 'th instance is incorrectly labeled, and to 0 otherwise. We similarly set the weight μ_j^t to be $\frac{1}{|\mathcal{M}^t|}$ for $j \in \mathcal{M}^t$ and to 0 otherwise. We abbreviate this variant as the *SimPerc* algorithm.

Soft Simultaneous Projections: The soft simultaneous projections variant uses the fact that each reduced problem has an analytic solution, yielding

$$\alpha_j^t = \min \left\{ C, \ell(\omega^t; (\mathbf{x}_j^t, y_j^t)) / \|\mathbf{x}_j^t\|^2 \right\}.$$

We independently assigns each α_j^t this optimal

defines a feasible solution $\omega = \sum_{t=1}^T \sum_{j=1}^{k_t} \lambda_{t,j} y_j^t \mathbf{x}_j^t$ and a corresponding selection of the slack variables.

Clearly, the optimization problem given by Eq. (5) depends on all the examples from the first trial through time step T and thus can only be solved in a 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 of rounds occurring later than t . As we show next, we use this primal-dual view to derive the skeleton algorithm from Fig. 2 by finding a new feasible solution for the dual problem on every trial. Formally, the instantaneous dual problem, given by Eq. (2), is equivalent (after omitting an additive constant) to the following constrained optimization problem,

$$\max_{\lambda} \mathcal{D}(\lambda_1, \dots, \lambda_{t-1}, \lambda, \mathbf{0}, \dots, \mathbf{0}) \quad \text{s.t.} \quad \lambda \geq \mathbf{0}, \quad \sum_{j=1}^{k_t} \lambda_j \leq C. \quad (6)$$

That is, the instantaneous dual problem is obtained from $\mathcal{D}(\lambda_1, \dots, \lambda_T)$ by fixing $\lambda_1, \dots, \lambda_{t-1}$ to the values set in previous rounds, forcing λ_{t+1} through λ_T to the zero vectors, and choosing a feasible vector for λ_t . Given the set of dual variables $\lambda_1, \dots, \lambda_{t-1}$ it is straightforward to show that the prediction vector used on trial t is $\omega^t = \sum_{s=1}^{t-1} \sum_j \lambda_{s,j} y_j^s \mathbf{x}_j^s$. Equipped with these relations and omitting constants which do not depend on λ_t Eq. (6) can be rewritten as,

$$\max_{\lambda_1, \dots, \lambda_{k_t}} \sum_{j=1}^{k_t} \lambda_j - \frac{1}{2} \left\| \omega^t + \sum_{j=1}^{k_t} \lambda_j y_j^t \mathbf{x}_j^t \right\|^2 \quad \text{s.t.} \quad \forall j : \lambda_j \geq 0, \quad \sum_{j=1}^{k_t} \lambda_j \leq C. \quad (7)$$

Since the problems defined by Eq. (7) and Eq. (2) are equivalent, weighing the variable $\alpha_1^t, \dots, \alpha_{k_t}^t$ by $\mu_1^t, \dots, \mu_{k_t}^t$ also yields a feasible solution for the problem defined in Eq. (6), namely $\lambda_{t,j} = \mu_j^t \alpha_j^t$. We now tie all of these observation together by using the weak-duality theorem (see for instance [8]). Our first bound is given for the the SimPerc algorithm.

Theorem 1. *Let $(X^1, Y^1), \dots, (X^T, Y^T)$ be a sequence of examples where $X^t = \{\mathbf{x}_j^t\}_{j=1}^{k_t}$ and $Y^t = \{y_j^t\}_{j=1}^{k_t}$. Assume that for all t and j the norm of an instance \mathbf{x}_j^t is at most R . Then, for any $\omega^* \in \mathbb{R}^n$ the number of trials on which the prediction of SimPerc is imperfect is at most,*

$$\frac{\frac{1}{2} \|\omega^*\|^2 + C \sum_{t=1}^T \ell(\omega^*; (X^t, Y^t))}{C - \frac{1}{2} C^2 R^2}.$$

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, denoted $\mathcal{P}(\omega^*, \xi^*)$. In particular, the solution obtained at the end of trial T is dual feasible, and thus $\mathcal{D}(\lambda_1, \dots, \lambda_T) \leq \mathcal{P}(\omega^*, \xi^*)$. We now rewrite the left hand-side of the above equation as the following sum,

$$\mathcal{D}(\mathbf{0}, \dots, \mathbf{0}) + \sum_{t=1}^T [\mathcal{D}(\lambda_1, \dots, \lambda_t, \mathbf{0}, \dots, \mathbf{0}) - \mathcal{D}(\lambda_1, \dots, \lambda_{t-1}, \mathbf{0}, \dots, \mathbf{0})]. \quad (8)$$

Note that $\mathcal{D}(\mathbf{0}, \dots, \mathbf{0})$ equals 0. Therefore, denoting by Δ_t the difference of two consecutive dual objective values, $\mathcal{D}(\lambda_1, \dots, \lambda_t, \mathbf{0}, \dots, \mathbf{0}) - \mathcal{D}(\lambda_1, \dots, \lambda_{t-1}, \mathbf{0}, \dots, \mathbf{0})$, we get that $\sum_{t=1}^T \Delta_t \leq \mathcal{P}(\omega^*, \xi^*)$. We now turn to bounding each Δ_t from below. First, note that if the prediction on trial t is perfect ($\mathcal{M}^t = \emptyset$) then SimPerc sets λ_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 ω^t we get that $\omega^t = \sum_{s < t} \sum_{j=1}^{k_s} \lambda_{s,j} y_j^s \mathbf{x}_j^s$. We now rewrite Δ_t as follows,

$$\Delta_t = \sum_{j=1}^{k_t} \lambda_{t,j} - \frac{1}{2} \left\| \omega^t + \sum_{j=1}^{k_t} \lambda_{t,j} y_j^t \mathbf{x}_j^t \right\|^2 + \frac{1}{2} \|\omega^t\|^2. \quad (9)$$

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. (9) and write,

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

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 ,

$$\Delta_t \geq \sum_{j=1}^{k_t} \mu_j^t \left[\alpha_j^t - \frac{1}{2} \|\omega^t + \alpha_j^t y_j^t \mathbf{x}_j^t\|^2 + \frac{1}{2} \|\omega^t\|^2 \right] . \quad (10)$$

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. (10) can be further simplified and written as,

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

We expand the norm in the above equation and obtain that,

$$\Delta_t \geq \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2} \|\omega^t\|^2 - C y_j^t \omega^t \cdot \mathbf{x}_j^t - \frac{1}{2} C^2 \|y_j^t \mathbf{x}_j^t\|^2 + \frac{1}{2} \|\omega^t\|^2 \right] . \quad (11)$$

The set \mathcal{M}^t consists of indices of instances which have been incorrectly classified. Thus, $y_j^t(\omega^t \cdot \mathbf{x}_j^t) \leq 0$ for every $j \in \mathcal{M}^t$. Therefore, Δ_t can further be bounded from below as follows,

$$\Delta_t \geq \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2} C^2 \|y_j^t \mathbf{x}_j^t\|^2 \right] \geq \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2} C^2 R^2 \right] = C - \frac{1}{2} C^2 R^2 , \quad (12)$$

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 $\Delta_t \geq C - \frac{1}{2} C^2 R^2$, while in perfect trials where no mistake is made $\Delta_t = 0$. Putting all the inequalities together we obtain the following bound,

$$\left(C - \frac{1}{2} C^2 R^2 \right) \epsilon \leq \sum_{t=1}^T \Delta_t = \mathcal{D}(\lambda_1, \dots, \lambda_T) \leq \mathcal{P}(\omega^*, \xi^*) , \quad (13)$$

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

The ConProj algorithm updates the same set of dual variables as the SimPerc algorithm, but selects α_j^t to be the optimal solution of Eq. (3). Thus, the value of Δ_t attained by the ConProj algorithm is never lower than the value potentially attained by the SimPerc algorithm. The following corollary is a direct consequence of this observation.

Corollary 1. *Using the same definitions of Thm. 1. Then, for any $\omega^* \in \mathbb{R}^n$ the number of trials on which the prediction of ConProj is imperfect is at most,*

$$\frac{\frac{1}{2} \|\omega^*\|^2 + C \sum_{t=1}^T \ell(\omega^*; (X^t, Y^t))}{C - \frac{1}{2} C^2 R^2} .$$

We note that the predictions of the SimPerc algorithm do not depend on the specific value of C , thus we may optimally set C to $\frac{1}{R^2}$. The bound attained in Thm. 1 now becomes.

$$2R^2 \cdot \left[\frac{1}{2} \|\omega^*\|^2 + C \sum_{t=1}^T \ell(\omega^*; (X^t, Y^t)) \right] .$$

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 only on classification errors and disregarding margin errors. While this approach ensures a substantial increase of the dual, in practice it appears to be a double edged sword as the SimProj algorithm performs empirically better.

username	k	m	SimProj	ConProj	SimPerc	Max-SP	Max-MP	Mira
beck-s	101	1973	50.0	55.2	55.9	56.6	63.8	63.7
farmer-d	25	3674	27.4	30.3	30.7	30.0	28.6	31.8
kaminski-v	41	4479	43.1	47.8	47.0	49.5	49.6	47.3
kitchen-l	47	4017	42.9	47.0	49.0	48.0	54.9	52.6
lokay-m	11	2491	18.8	25.3	25.3	23.0	25.4	25.3
sanders-r	30	1190	20.7	25.6	23.2	23.8	36.3	34.1
williams-w3	18	2771	4.2	5.0	5.4	4.2	5.8	5.9

Table 1: The percentage of online mistakes of the three variants compared to Max-Update (Single prototype (SP) and Multi prototype (MP)) and the Mira algorithm. Experiments were performed on seven users of the Enron data set.

5 Experiments

In this section we describe experimental results in order to demonstrate some of the merits of our algorithms. We tested performance of the three variants described in Sec. 3 on a multiclass categorization task and compared them to previously studied algorithms for multiclass categorization. We compared our algorithms to the single-prototype and multi-prototype Max-Update algorithms from [9] and to the Mira algorithm [2]. The experiments were performed on the task of email classification using the Enron email dataset (Available at http://www.cs.cmu.edu/~enron/enron_mail_030204.tar.gz). The learning goal was 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 $\{1, \dots, k\}$. 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 one as a separate online learning problem. We represented each email message as a vector with a component for every word in the corpus. At each trial, and for each class r , we constructed class-dependent vectors as follows. We set $\phi_j(\mathbf{x}^t, r)$ to 1, if the j 'th word appeared in the message, and in a fifth of the messages previously assigned to folder r . Similarly, we set $\phi_j(\mathbf{x}^t, r)$ to -1 , if the 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. Next, we employed the mapping described in Sec. 2, and defined a set of $k - 1$ instances for each message as follows. Denote the relevant class by r , then for every irrelevant class $s \neq r$, we define an instance $\mathbf{x}_s^t = \phi(\mathbf{x}^t, r) - \phi(\mathbf{x}^t, s)$ and set its label to 1. All these instances were gathered into a single set $X^t = \{\mathbf{x}_s^t\}_{s \neq r}$ and were provided to the algorithm in trial t .

The results of the experiments are summarized in Table 1. It is apparent that the SimProj algorithm outperforms all other algorithms. The performances of SimPerc and ConProj are comparable with no obvious winner. It is worth noting that the Mira algorithm finds the optimum of a projection problem at each trial while our algorithms only find an approximate solution. However, Mira employs a different approach in which there is a single input instance (instead of the set X^t) and constructs multiple predictors (instead of a single vector ω). Thus, Mira employs a larger hypothesis space which is more difficult to learn in online settings. In addition, by employing a single vector representation of the email message, Mira cannot benefit from principled feature selection which results from class-dependent features. It is also obvious that the simultaneous projection variants, while remaining simple to implement, consistently outperform the Max-Update technique which is commonly used in online multiclass classification. In Fig. 3 we plot the cumulative number of mistakes as a function of the trial number for 6 of the 7 users. (Due to space limitations, we omitted the user william-w3 who simply classifies 85 percent of his mails in just two folder and constitutes an easy classification task.) The graphs clearly indicate the high correlation between the *SimPerc* and *ConProj* variants, while indicating the superiority of the *SimProj* variant.

6 Extensions and discussion

We presented a new approach to online categorization with complex output structure. Our algorithm breaks 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 followed by averaging the solutions automatically adheres with this constraint and hence constitute a feasible

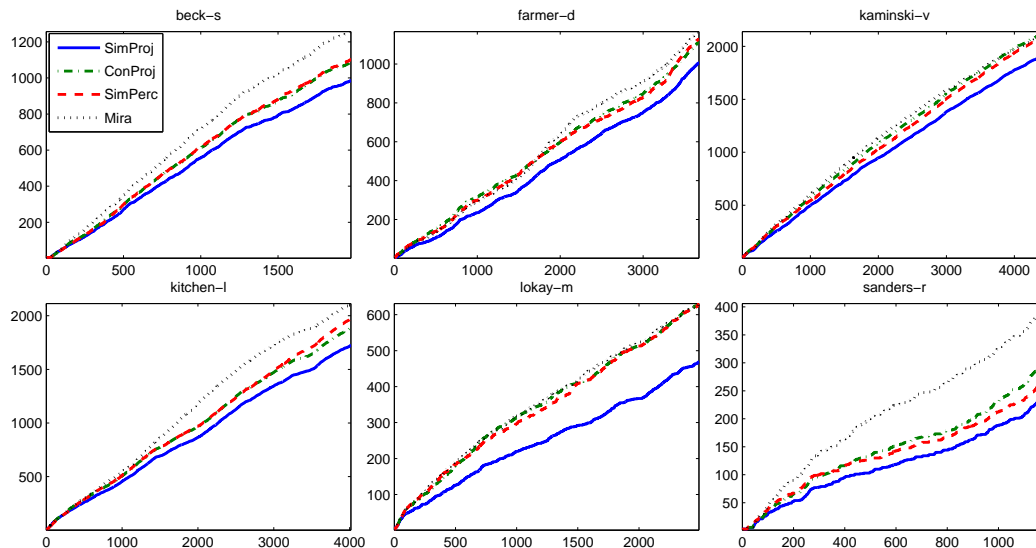


Figure 3: The cumulative number of mistakes as a function of the number of trials.

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. We are currently exploring the usage of this extension in complex decision problems with multiple structural constraints. Another possible extension is to replace the squared norm regularization with other twice differentiable penalty functions. Algorithms of this more general framework still attain similar mistake bounds and are easy to implement so long as the induced individual problems are efficiently solvable. A particularly interesting case is obtained when setting the penalty to the relative entropy. In this case we obtain a generalization of the Winnow and the EG algorithms [10, 11] for complex classification problems. Another interesting direction is the usage of simultaneous projections for problems with more constrained structured output by decoupling, for instance, the constraints imposed in max-margin networks [3].

References

- [1] 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.
- [2] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.
- [3] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.
- [4] I. Tsochantaris, 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.
- [5] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.
- [6] R.E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 32(2/3), 2000.
- [7] 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).)
- [8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [9] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7, Mar 2006.
- [10] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [11] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.