A Family of Large Margin Linear Classifiers and Its Application in Dynamic Environments

Jianqiang Shen
Pearson Knowledge Technologies
299 S. California Ave, Palo Alto, CA94306
jianqiang.shen@pearson.com

Thomas G. Dietterich
1148 Kelley Engineering Center
Oregon State University, Corvallis, OR97331
tgd@eecs.oregonstate.edu

Abstract

Real-time problems prediction problems pose a challenge to machine learning algorithms because learning must be fast, the set of classes may be changing, and the relevance of some features to each class may be changing. To learn robust classifiers in such nonstationary environments, it is essential not to assign too much weight to any single feature. We address this problem by combining regularization mechanisms with online large margin learning algorithms. We prove bounds on their error and show that removing features with small weights has little influence on prediction accuracy, suggesting that these methods exhibit feature selection ability. We show that such regularized learning algorithms automatically decrease the influence of older training instances and focus on the more recent ones. This makes them especially attractive in dynamic environments. We evaluate our algorithms through experimental results on real data sets and through experiments with an online activity recognition system. The results show that these regularized large-margin methods adapt more rapidly to changing distributions and achieve lower overall error rates than state-of-the-art methods.

Keywords: Classification, online learning, feature selection, activity recognition, non-stationary environments.

1 Introduction

When applying machine learning to real-world problems such as web page categorization and activity recognition, efficiency and non-stationarity are important issues. Popular web sites, such as blogs and newspapers, continually generate large numbers of novel documents. Learning over such web-scale data is very expensive, even with thousands of machines running in parallel [24]. In addition, topics of interest change rapidly – a topic with millions of visits last year might be completely ignored this year.

Similar challenges arise when intelligent personal assistants seek to track desktop activities (e.g., [18, 17, 26, 30]). Each time the user provides feedback (either implicit or explicit) about a prediction, the assistant needs to update its predictor. This update must be very efficient in order for such an assistant to be usable. User activity patterns change over time, as a result of changes in the mix of projects and deadlines, which again raises the problem of non-stationarity.

In this paper, we explore learning algorithms that are able to efficiently handle large-scale data sets and rapidly adapt to changes in the set of categories, their definitions, and their relative frequencies.

Online learning algorithms are algorithms that consume a constant amount of storage and incrementally update the classifier in a constant amount of time for each new training example. In the stationary case, the ultimate accuracy of such algorithms is limited by the total number of training examples. However, in the non-stationary case, only the more recent training examples within each class are relevant to recognizing that class. The greater the degree of non-stationarity (i.e., the faster the problem is changing), the smaller the effective number of training examples. As a consequence, the risk of overfitting increases, and it is important to prevent overfitting through some form of regularization or feature selection. This is particularly true for text classification problems where there are thousands of candidate features (words).

In this paper, we design and evaluate efficient large margin learning algorithms by combining regularization mechanisms with online updates. Regularization has been shown to be effective for batch learning algorithms when learning from data with many irrelevant features [33, 15]. The regularization penalty shrinks the classifier weights towards zero and has the effect of controlling the variance of the learned model. Appropriate regularization can generally reduce over-fitting by trading off a small increase in bias for a large reduction in variance. Compared with feature selection, regularization is a continuous process that shrinks the influence of some features. Because it can be implemented in an online algorithm, unlike standard feature selection methods, it is more suitable for nonstationary data. Unlike other weight-shrinking online learning algorithms [16, 21], our al-

gorithms penalizes the model complexity without compromising the margin of the training instances. This paper investigates both L1 and L2 regularization for online updates. We analyze the characteristics of the regularization mechanism in online learning settings.

The regularization penalty drives the weights of many features towards zero. The theoretical analysis shows that ignoring features with small weights has little influence on the prediction accuracy. This feature selection effect can also explain why regularized online learning is usually more accurate, as confirmed by our experiments. For real-world online learning problems, the distribution generating the data is usually changing as the time passes. A very discriminative feature can rapidly become less useful or even useless. By avoiding over-weighting a feature, our regularized methods can shift to the right model more quickly when the data changes. We also show that the L2 regularized learning method has another property appropriate for dynamic environments – it automatically shrinks the influence of older training instances and pays more attention to more recent ones.

We present an application of our algorithms to an intelligent activity management system, the TaskTracer system [10, 30]. TaskTracer helps users organize and retrieve information based on user activities. It collects various timestamped user interactions (such as file open, save, text selection, copy/paste, window focus, web navigation, email read/send and so on) in Microsoft Office (Word, Excel, PowerPoint, Outlook), text and pdf files, Internet Explorer, and the Windows operating system. Each interaction generates events which are stored in a database. TaskTracer associates with each activity the set of resources accessed when performing that activity. It employs this data to configure the desktop to assist users in organizing and re-finding information. TaskTracer requires the user to explicitly declare the current activity in order to correctly associate resources with activities. This approach fails when the user is interrupted (e.g., by a phone call, instant message). The user typically changes documents, web pages, and so on without remembering to first inform TaskTracer. To address this problem, we designed and implemented an activity recognition component by applying a variation of the learning algorithms described in this paper [29].

This paper is organized as follows. We begin with an introduction to online learning and a discussion of our motivation. We then derive our regularized large margin algorithms and present their theoretical analysis. We show experimental results on real data sets to evaluate the performance of our algorithms. We present an application in the activity recognition component of TaskTracer. We conclude the paper with a discussion of future work.

2 Online Algorithms and Dynamic Environments

An online learning algorithm processes instances in sequence. In iteration t, the algorithm receives instance $\mathbf{x}_t \in \mathbb{R}^n$ and makes a prediction with its current learned function f_t. Then it receives y_t, the correct label of $\mathbf{x}_t$, and computes the update condition $\mathcal{C}$. If $\mathcal{C}$ is true, it updates f_t to produce f_{t+1} so that a requirement set $\mathcal{R}$ is satisfied. The goal is to minimize the online prediction error of a single pass over all instances [2]. Online learning algorithms can be characterized based on their choices of $\mathcal{C}$ and $\mathcal{R}$ [28, 22, 13, 23, 21, 7, 6]. We focus on the binary class problem where $y_t \in \{+1, -1\}$ and f_t is a linear classifier: $f_t = \mathbf{w}_t \cdot \mathbf{x}_t$. The results can easily be generalized to multiclass problems. We consider only additive update algorithms.

The term $y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ is generally referred to as the *margin*. Enforcing a large margin can often improve prediction accuracy. In this paper, we consider algorithms that perform a *Passive-Aggressive* (*PA*) update [6] when the classifier fails to correctly classify an instance with a functional margin of 1. The PA update modifies the learned function subject to two constraints: (a) the correct label should have a functional margin of 1, and (b) the change to the weights should be minimal in order to reduce fluctuations. The PA update sets the new weight vector $\mathbf{w}_{t+1}$ to be the solution to the following constrained optimization problem,

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|_2^2 \\ \text{s.t.} & \\ & y_t(\mathbf{w}\cdot\mathbf{x}_t) \geq 1. \end{aligned}$$

Most online learning algorithms have no limitation on the size of the feature weights. Consequently, some weights can grow to be quite large. This is inappropriate in dynamic environments, where features that were important at one time may become unimportant at later times. For example, let's consider the problem of sentiment prediction where the goal is to predicts whether a product review is positive. Consider an imaginary product, the *iLearn*. Suppose that when it first appears on the market, it gets many positive reviews, because of its novel functionality. Hence, the word *iLearn* is a good indicator of positive sentiment and receives a large weight. But then suppose that a serious problem is discovered with the *iLearn*. Then the word *iLearn* immediately changes from predicting positive sentiment to predicting negative sentiment. However, since *iLearn* received a large positive weight during the early phases, it may take standard learning algorithms a long time to respond to the change. In particular, standard methods such as Naive Bayes would not realize that *iLearn* is now predictive of negative sentiment until it had seen as many negative examples as it had previously seen positive examples. In the mean time, the algorithm will make many prediction errors. One way to

avoid this problem is to avoid assigning too much weight to any one feature.

When constructing classifiers over high-dimensional data sets, we face problem of over-fitting. A common strategy for addressing this issue is to first perform a feature selection step. Standard feature selection methods [35] adopt the batch approach and thus are inappropriate for online learning. Some feature selection methods have been designed for the online setting [14, 9], but they have two shortcomings. First, they assume an adversarial environment and take a worse-case approach. Thus the performance is usually suboptimal in the normal case. Second, they usually must solve a difficult optimization problem that lacks a closed-form solution. In this paper, we address the over-fitting problem by applying regularization. We show that our algorithms have feature selection ability and can improve over non-regularized algorithms.

We say that an instance is *active* if it triggers an update. Online learning algorithms typically set the initial weight vector to the zero vector and do updates of the form $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$ where τ_t is the learning rate determined by the learning algorithm. Thus, $\mathbf{w}_t$ is a linear combination of the active instances, and the newer active instances play the same role as the older active instances. We show that for certain kinds of regularized online learning, the updates have the form $\mathbf{w}_{t+1} = \frac{1}{Z_t}\mathbf{w}_t + \tau_t y_t \mathbf{x}_t$, where $Z_t \geq 1$. Thus, the coefficients of those active instances appearing earlier shrink and have less influence as additional instances are received. Note that this is exactly the opposite of standard stochastic gradient descent algorithms, which decrease τ (usually as $1/t$), and hence place more weight on older instances and less weight on more recent ones [27].

3 Regularized Online Learning of Linear Classifiers

We investigate two kinds of regularization. The first employs a penalty in the objective function, and the second one places explicit norm requirement in the constraints.

3.1 Online Learning with a Regularized Objective.

Let α be a constant controlling the shrinkage rate. We can shrink the norm of the weight vector towards zero by adding a penalty in the objective function:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|_2^2 + \frac{\alpha}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to } y_t(\mathbf{w}\cdot\mathbf{x}_t) \geq 1, \tag{3.1}$$

We denote the hinge loss [15] at iteration t as ℓ_t. This gives a simple closed-form update:

LEMMA 3.1. *Problem 3.1 has the closed-form solution* $\mathbf{w}_{t+1} = \frac{1}{1+\alpha}(\mathbf{w}_t + \tau_t y_t \mathbf{x}_t)$, *where* $\tau_t = \frac{\ell_t + \alpha}{\|\mathbf{x}_t\|_2^2}$.

Proof. The Lagrangian of the optimization problem in Problem 3.1 is

$$L(\mathbf{w}, \tau) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|_2^2 + \frac{\alpha}{2}\|\mathbf{w}\|_2^2 + \tau(1 - y_t(\mathbf{w}\cdot\mathbf{x}_t)), \tag{3.2}$$

where $\tau \geq 0$ is the Lagrange multiplier. Differentiating this Lagrangian with respect to the elements of $\mathbf{w}$ and setting the partial derivatives to zero gives

$$\mathbf{w} = \frac{1}{1+\alpha}\mathbf{w}_t + \frac{\tau}{1+\alpha} y_t \mathbf{x}_t. \tag{3.3}$$

Replacing $\mathbf{w}$ in Eq 3.2 with Eq 3.3, the Lagrangian becomes

$$L(\tau) = \frac{1}{2}\left\|\frac{\tau}{1+\alpha} y_t \mathbf{x}_t - \frac{\alpha}{1+\alpha}\mathbf{w}_t\right\|_2^2 + \frac{\alpha}{2}\left\|\frac{\tau}{1+\alpha} y_t \mathbf{x}_t + \frac{1}{1+\alpha}\mathbf{w}_t\right\|_2^2 + \tau\left(1 - \frac{y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{1+\alpha} - \frac{\tau\|\mathbf{x}_t\|_2^2}{1+\alpha}\right).$$

By setting the derivative of this with respect to τ to zero, we obtain

$$1 - \frac{\tau}{1+\alpha}\|\mathbf{x}_t\|_2^2 - \frac{y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{1+\alpha} = 0$$
$$\Rightarrow \quad \tau = \frac{1 - y_t(\mathbf{w}_t\cdot\mathbf{x}_t) + \alpha}{\|\mathbf{x}_t\|_2^2}. \qquad \blacksquare$$

We will refer to this algorithm as the **objective-regularized algorithm**, because it places the regularization in the objective function of the optimization problem. This algorithm combines (a) largin margin fitting of the data (the constraints) with (b) minimizing the change in the weights (first term of the objective) with (c) minimizing the magnitude of the weights (second term of the objective). The first term of the objective could also be viewed as being the "memory term", since it seeks to remember the previous weights, while the second term in the object can be viewed as the "forgetting term", since it serves to shrink the weights. Hence, the algorithm can be viewed as balancing the competition between remembering and forgetting, as controlled by α. Notice that in all cases, our algorithm must fit the most recent training example with a functional margin of 1.

A few other online learning algorithms attempt to shrink weights towards zero [16, 21]. These algorithms trade off fitting the data against simplicity of the learned model. Their updates do not directly ensure a large margin. Consequently, their shrinking mechanisms have to sacrifice the fitting to the training data. Experimental results show that our algorithm gives much higher accuracy.

Since we are handling nonstationary problems, the best hypothesis at each iteration might be changing. Let us define the *optimal algorithm* to be one that performs the minimal number of updates and does not update its hypothesis unless our algorithm updates the hypothesis—that is, it may perform fewer updates by skipping some of the updates that our

algorithm makes. We will show that our regularized algorithm is competitive with the optimal algorithm, as long as the change between two consecutive optimal hypotheses is not extremely dramatic.

Let $\mathbf{u}_0, .., \mathbf{u}_T \in \mathbb{R}^n$ be the sequence of weight vectors chosen by the optimal algorithm. We will specify that $\mathbf{u}_0 = \mathbf{0}$, the zero vector. Let ℓ_t^* denote the loss of $\mathbf{u}_t$ at iteration t. Assume that the norm is bounded for each instance $\mathbf{x}_t$, i.e., $\|\mathbf{x}_t\|_2 \leq R$. Let I_t be the *active set* at iteration t for our algorithm. An instance $\mathbf{x}_i$ is in I_t iff $i < t$ and it triggers an update by our algorithm. The following theorem provides an error bound of our algorithm.

THEOREM 3.1. *Assume that there exists an optimal sequence of vectors $\mathbf{u}_0, ..., \mathbf{u}_T \in \mathbb{R}^n$ such that $\|\mathbf{u}_t\|_2 = D$, $\ell_t^* = 0$ for all t, $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_2 \leq \mu$ and μ satisfies $g = \frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} > 0$. Given $\max_t \|\mathbf{w}_t\|_2 \leq \beta$, the number of prediction mistakes made by the objective-regularized algorithm is bounded by $m \leq \frac{D^2}{g}$.*

Proof. Let $\Delta_t = \|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2$. We can prove the bound by lower and upper bounding $\sum_t \Delta_t$. Since $\mathbf{w}_0$ is a zero vector and the norm is non-negative, $\sum_t \Delta_t = \|\mathbf{w}_0 - \mathbf{u}_0\|_2^2 - \|\mathbf{w}_T - \mathbf{u}_T\|_2^2 \leq \|\mathbf{w}_0 - \mathbf{u}_0\|_2^2 = D^2$.

Obviously, $\Delta_t \neq 0$ only if $t \in I_T$, where I_t is the active set of our algorithm at iteration t. We will only consider this case here. Let $\mathbf{w}'_t = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$, $\mathbf{w}_{t+1} = \frac{1}{1+\alpha}\mathbf{w}'_t$. Δ_t can be rewritten as

$$\begin{aligned}
&(\|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}'_t - \mathbf{u}_t\|_2^2) \\
&+(\|\mathbf{w}'_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2) \\
&+(\|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2) = \delta_t + \psi_t + \epsilon_t.
\end{aligned}$$

We will lower bound δ_t, ψ_t and ϵ_t.

For δ_t, we have

$$\begin{aligned}
\delta_t =& -2\tau_t y_t \mathbf{x}_t \cdot (\mathbf{w}_t - \mathbf{u}_t) - \|\tau_t y_t \mathbf{x}_t\|_2^2 \\
\geq& 2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|_2^2 .
\end{aligned}$$

Plugging the definition of τ_t and considering $\ell_t \geq 1$ get

$$\delta_t \geq \frac{2\ell_t^2 + 2\ell_t\alpha}{\|\mathbf{x}_t\|_2^2} - \frac{\ell_t^2 + 2\ell_t\alpha + \alpha^2}{\|\mathbf{x}_t\|_2^2} \geq \frac{1-\alpha^2}{R^2}. \tag{3.4}$$

For ψ_t, we have

$$\begin{aligned}
\psi_t =& -2\mathbf{w}'_t \cdot (\mathbf{u}_t - \mathbf{u}_{t+1}) \\
\geq& -2\|\mathbf{w}'_t\|_2 \|\mathbf{u}_t - \mathbf{u}_{t+1}\|_2 \geq -2(1+\alpha)\mu\beta.
\end{aligned} \tag{3.5}$$

For ϵ_t, we have

$$\epsilon_t = (1 - \frac{1}{(1+\alpha)^2}) \|\mathbf{w}'_t\|_2^2 - 2(1 - \frac{1}{1+\alpha})\mathbf{w}'_t \cdot \mathbf{u}_{t+1}.$$

Using the fact that $\|\mathbf{u} - \mathbf{v}\|_2^2 \geq 0$ which is equivalent to $\|\mathbf{u}\|_2^2 - 2\mathbf{u}\cdot\mathbf{v} \geq -\|\mathbf{v}\|_2^2$, we get

$$\begin{aligned}
&(1 - \frac{1}{(1+\alpha)^2}) \|\mathbf{w}'_t\|_2^2 - 2(1 - \frac{1}{1+\alpha})\mathbf{w}'_t \cdot \mathbf{u}_{t+1} \\
\geq& -\frac{1 - \frac{1}{1+\alpha}}{1 + \frac{1}{1+\alpha}} \|\mathbf{u}_{t+1}\|_2^2 = -\frac{\alpha D^2}{2+\alpha}.
\end{aligned} \tag{3.6}$$

Using Eq 3.4, 3.5 and 3.6, we obtain

$$\sum_{t=1}^{T} \Delta_t \geq m \left(\frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} \right).$$

Applying $\sum_t \Delta_t \leq D^2$ gives

$$m \left(\frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} \right) \leq D^2. \tag{3.7}$$

Since $g = \frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} > 0$, we get the result in the theorem. ∎

Note that the error bound increases as μ (the bound on the size of the optimal weight changes) increases. This confirms the intuition that a learning problem with dramatically changing concepts is difficult, even for a regularized online learning algorithm.

3.2 Online Learning with a Norm Constraint. The objective-regularized algorithm keeps shrinking the weights even when the weights have become quite small. This could hurt prediction accuracy. We can instead only shrink the weights when they get too large by enforcing a norm constraint:

$$\begin{aligned}
\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 \\
\text{subject to } \ y_t(\mathbf{w}\cdot\mathbf{x}_t) \geq 1 \text{ and } \|\mathbf{w}\|_2 \leq \beta.
\end{aligned} \tag{3.8}$$

This leads to the following simple closed-form update:

LEMMA 3.2. *Problem 3.8 has the closed-form solution $\mathbf{w}_{t+1} = \frac{1}{Z_t}(\mathbf{w}_t + \tau_t y_t \mathbf{x}_t)$, where $Z_t = \max\left\{1, \sqrt{\frac{\|\mathbf{w}_t\|_2^2\|\mathbf{x}_t\|_2^2 - (\mathbf{w}_t\cdot\mathbf{x}_t)^2}{\beta^2\|\mathbf{x}_t\|_2^2 - 1}}\right\}$, $\tau_t = \frac{\ell_t + Z_t - 1}{\|\mathbf{x}_t\|_2^2}$.*

A detailed proof is presented in the Appendix. We will refer to this algorithm as the **L2-norm constrained algorithm**, because it places an L2 norm constraint on the weight vector. This algorithm performs the normal passive-aggressive update until the norm of $\mathbf{w}_t$ exceeds β. It then shrinks the weights enough to ensure that the β constraint is satisfied. Our experiments show that this approach is slightly more accurate than the objective-regularized algorithm.

It is easy to show that both the objective-regularized and the L2 norm constrained algorithms are rotationally

invariant. Let $\mathcal{M} = \{M \in \mathbb{R}^{n\times n} | MM' = M'M = I, |M| = 1\}$ be the class of rotational matrices, where I is the identity matrix. Given a learning algorithm L, we say it is *rotational invariant* [25] if for any training set S, rotational matrix $M \in \mathcal{M}$, and test example x, we have $L[S, x] = L[MS, Mx]$, where $L[S, x]$ is the predicted label of x resulting from using L to train on S.

LEMMA 3.3. *The learning algorithm solving Problem 3.1 and the algorithm solving Problem 3.8 are rotationally invariant.*

Proof. We focus on the Problem 3.1. The proof can be similarly applied to Problem 3.8. We show that if L trained with S outputs $\mathbf{w}$, then L trained with MS outputs the weight vector $M\mathbf{w}$ by induction on the size of S.

Let $L[S]$ denote the weight vector returned by L trained with S.

When $|S| = 0$, both weight vectors are zero vectors. Thus, the claim is true and the score functions will be the same since they always return 0.

Assume when $|S| = k$, $L[MS] = ML[S]$. Now, consider $|S| = k + 1$. Let $S = S' \cup \{x_{k+1}\}$ and $MS = (MS') \cup \{Mx_{k+1}\}$. We know $L[MS'] = ML[S']$, since the size of S' is k. Since we are using the linear product as the score function, for L trained with MS' we have the score for Mx_{k+1}: $L[MS'] \cdot (Mx_{k+1}) = (ML[S']) \cdot (Mx_{k+1}) = L[S'] (MM) x_{k+1} = L[S'] \cdot x_{k+1}$. Thus the prediction of Mx_{k+1} given by L with MS' is the same with the prediction of x_{k+1} given by L with S'.

If there is no need to update the weights, we obviously have $L[MS] = ML[S]$. If we need to update the weights, the update will be $L[MS] = \frac{1}{1+\alpha}(L[MS'] + \tau'_t y_t (M\mathbf{x}_{k+1}))$, where $\tau'_t = \frac{\ell'+\alpha}{\|M\mathbf{x}_{k+1}\|^2} = \frac{\ell+\alpha}{\|\mathbf{x}_{k+1}\|^2} = \tau_t$. Thus $L[MS] = M\frac{1}{1+\alpha}(L[S'] + \tau_t y_t (\mathbf{x}_{k+1})) = ML[S]$.

In summary, $L[MS] = ML[S]$ given any data set. Since we are using the linear product as the score function, we always have $L[S, x] = L[MS, Mx]$. ∎

Ng [25] shows that rotationally invariant algorithms can require a large number of training instances to learn a simple model when there are many irrelevant features. In such situations, learning algorithms with L1 regularization usually learn more quickly. Thus, it is worth exploring the following L1 norm constrained learning algorithm:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|_2^2 \quad \text{subject to } y_t(\mathbf{w}\cdot\mathbf{x}_t) \geq 1 \text{ and } \|\mathbf{w}\|_1 \leq \beta. \tag{3.9}$$

This can be transformed into a quadratic programming problem and solved with an off-the-shelf quadratic programming package. Since the L1 norm constraint is still convex, it is guaranteed to find the global optimum. There is a discontinuity in the gradient of L1 with respect to w_i at $w_i = 0$. This tends to force a subset of weights to be exactly zero [33], so that the learned weight vector is sparse.

The following theorem provides an error bound for the online learning algorithms with L2 or L1 norm constraints:

THEOREM 3.2. *Assume that there exists an optimal sequence of vectors $\mathbf{u}_0, ..., \mathbf{u}_T \in \mathbb{R}^n$ such that $\|\mathbf{u}_t\|_2 = D \leq \beta$, $\ell_t^* = 0$ for all t, $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_2 \leq \mu$ and μ satisfies $\frac{1}{R^2} - 2\mu\beta > 0$, then the number of errors made by the algorithm with the L2 norm constraint is bounded by $m \leq \frac{R^2D^2}{1-2\mu\beta R^2}$. Similarly, if there exists an optimal vector sequence such that $\|\mathbf{u}_t\|_1 = D \leq \beta$ and $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_\infty \leq \mu$, then the number of errors made by the algorithm with the L1 norm constraint is bounded by $m \leq \frac{R^2D^2}{1-2\mu\beta R^2}$.*

Proof. We concentrate on the L2 norm case. The proof can be applied to the L1 case similarly. Let $\Delta_t = \|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2$. We can prove the bound by lower and upper bounding $\sum_t \Delta_t$. We know $\sum_t \Delta_t \leq D^2$.

We now lower bound Δ_t. We let $\Delta_t = (\|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_t\|_2^2) + (\|\mathbf{w}_{t+1} - \mathbf{u}_t\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2) = \gamma_t + \chi_t$.

For χ_t, we have $\chi_t = -2\mathbf{w}_{t+1} \cdot (\mathbf{u}_t - \mathbf{u}_{t+1}) \geq -2\mu\beta$.

It is obvious that $\|\mathbf{u}\|_1 = \sum_i |u_i| \geq |\sum_i u_i|$. Thus

$$\|(\mathbf{w}_{t+1} - \mathbf{w}_t)\mathbf{x}_t\|_1 \geq |y_t(\mathbf{w}_{t+1} - \mathbf{w}_t)\cdot\mathbf{x}_t| \geq 1. \tag{3.10}$$

Applying Holder's inequality, we have

$$\|(\mathbf{w}_{t+1} - \mathbf{w}_t)\mathbf{x}_t\|_1 \leq \|\mathbf{w}_{t+1} - \mathbf{w}_t\|_2 \|\mathbf{x}_t\|_2 \tag{3.11}$$

$$\|\mathbf{w}_{t+1} - \mathbf{w}_t\|_2 \geq \frac{1}{\|\mathbf{x}_t\|_2} \geq \frac{1}{R}. \tag{3.12}$$

Let $\mathcal{P}_S(\mathbf{w})$ denote the projection of $\mathbf{w}$ onto the convex set S. We know for any $\mathbf{u} \in S$, we have $\|\mathbf{w} - \mathbf{u}\|_2^2 - \|\mathcal{P}_S(\mathbf{w}) - \mathbf{u}\|_2^2 \geq \|\mathbf{w} - \mathcal{P}_S(\mathbf{w})\|_2^2$ [3].

We note that $\mathbf{w}_{t+1}$ is a projection of $\mathbf{w}_t$ onto the convex set $S = \{\mathbf{w} \in \mathbb{R}^n | loss(\mathbf{w}, (y_t, \mathbf{x}_t)) = 0 \text{ and } \|\mathbf{w}\|_2 \leq \beta\}$ for the L2 norm constrained algorithm. Since $\mathbf{u}_t \in S$,

$$\gamma_t \geq \|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2 \geq \frac{1}{R^2}. \tag{3.13}$$

Thus, we have

$$\sum_{i\in I_T} \left(\frac{1}{R^2} - 2\mu\beta\right) \leq D^2 \tag{3.14}$$

$$m \leq \frac{R^2D^2}{1 - 2\mu\beta R^2}. \tag{3.15}$$

Hence, we obtain the result in the theorem. ∎

Since $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_\infty \leq \|\mathbf{u}_t - \mathbf{u}_{t+1}\|_2$, the error bounds suggest that the L1 algorithm can tolerate more fluctuations of the best hypotheses than the L2 algorithm.

3.3 Regularization and Dynamic Environments. Our regularized algorithms have some characteristics especially desirable in dynamic environments.

The algorithms regularized with L2 norm automatically shrink the influence of the old observations and put more weight on the recent ones. Let I_t denote the active set at iteration t. Based on Lemma 3.1, the learned decision function of the objective-regularized algorithm at time t can be rewritten as

$$f_t(\mathbf{x}) = \text{sign}\left(\sum_{i\in I_t} \frac{\tau_i y_i}{(1+\alpha)^{|I_t - I_i|}} \mathbf{x}_i \cdot \mathbf{x}\right), \tag{3.16}$$

where $|I_t - I_i|$ is the number of active examples (and hence, the number of weight updates) from time $i+1$ up to time t.

Similarly, for the L2-norm constrained algorithm, we can rewrite the learned decision function at time t based on Lemma 3.2 as

$$f_t(\mathbf{x}) = \text{sign}\left(\sum_{i\in I_t} \frac{\tau_i y_i}{\prod_{j\in(I_t - I_i)} Z_j} \mathbf{x}_i \cdot \mathbf{x}\right). \tag{3.17}$$

These forms are interestingly similar to the Forgetron algorithm [8], an online kernel-based learning algorithm. The Forgetron performs the standard percetron update but controls the number of support vectors by removing the oldest support vector when the size of the support vector set is too large. Since removing a support vector may significantly change the hypothesis, it tries to "shrink" the weight of old support vectors. It multiplies the weights by $\phi_t \in (0,1]$ in each iteration. Our L2-regularized algorithms automatically shrink the weights of those support vectors and, hence, "forget" the old support vectors. This is critically important for handling non-stationary online learning problems.

Note also that these forms only involve inner products between training examples $\mathbf{x}_i$ and the new input example $\mathbf{x}$. This suggests that the method can be extended to non-linear decision boundaries by applying the kernel trick to replace the inner product with a function $K(\cdot,\cdot)$ that satisfies the Mercer conditions.

Theorem 3.2 also explains why it is important to shrink the norm of the weight vector towards zero and not to assign too much weight to any single feature in a dynamic environment. In order to be competitive with the optimal sequence of vectors, the norm of the learned vector has to satisfy $\frac{1}{R^2} - 2\mu\beta > 0$, or $\beta < \frac{1}{2\mu R^2}$. When the change between two consecutive optimal hypotheses becomes more dramatic (i.e., μ becomes larger), the valid value of β becomes even smaller. In such cases, we need to keep the norm of the weight vector small so that the learned hypothesis can adapt to the new environment quickly.

In Theorem 3.1, since we usually set α quite small, we can approximate g with $\frac{1}{R^2} - 2\mu\beta$ by ignoring α. This expression then matches the requirement in Theorem 3.2. Thus by analogy, when the optimal hypotheses change dramatically (i.e., when μ is large), we need to shrink the norm of our learned model more aggressively (keep β small) for the norm-penalized algorithms.

We cannot set α extremely large or set β extremely small, since we must ensure that the correct label can receive a positive score by a given margin. Thus the norm of the weight vector cannot be too small. But when possible, we should keep the norm of the learned weight vector small in order to improve the prediction precision in dynamic environments. To see this, note that the error bound $\frac{R^2D^2}{1-2\mu\beta R^2}$ in Theorem 3.2 is directly proportional to β, the norm of the weight vector. Ensuring a smaller β will lead to a smaller guaranteed error bound.

3.4 The Feature Selection Effect of Regularized Online Learning. Regularized online learning methods force many feature weights to be small. We now show that we can remove these features with small weights without hurting the accuracy too much. Assume that features are sorted in ascending order according to the absolute values of their weights. Suppose we remove the first k features so that $\sum_{i=1}^{k} w_i^2 < \sigma$ and $\sum_{i=1}^{k+1} w_i^2 \geq \sigma$. As long as σ is small, the number of errors will still be close to the original regularized method. For the objective-regularized algorithm, we have the following error bound:

THEOREM 3.3. *Assume that there exists an optimal sequence of weight vectors $\mathbf{u}_0, ..., \mathbf{u}_T \in \mathbb{R}^n$ such that $\|\mathbf{u}_t\|_2 = D$, $\ell_t^* = 0$ for all t, $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_2 \leq \mu$ and μ satisfies $h = \frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} - 2\sqrt{\sigma}D > 0$. Given $\max_t \|\mathbf{w}_t\|_2 \leq \beta$, then the number of prediction mistakes made by the objective-regularized algorithm which removes small weights is bounded by $m \leq \frac{D^2}{h}$.*

Proof. Let $\Delta_t = \|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2$. We can prove the bound by lower and upper bounding $\sum_t \Delta_t$. As the above, we know $\sum_t \Delta_t \leq D^2$.

Let $\mathbf{w}'_t = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$, $\mathbf{w}''_t = \frac{1}{1+\alpha}\mathbf{w}'_t$. We define vector $\mathbf{v}_t \in \mathbb{R}^n$ as

$$\mathbf{v}_{ti} = \begin{cases} \mathbf{w}''_{t\,i} & \text{if } \sum_{j=1}^{i} \mathbf{w}''^{\,2}_{t\,j} < \sigma \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

Then $\mathbf{w}_{t+1} = \mathbf{w}''_t - \mathbf{v}_t$. Δ_t can be rewritten as

$$\begin{aligned} &(\|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}'_t - \mathbf{u}_t\|_2^2) \\ &+ (\|\mathbf{w}'_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2) \\ &+ (\|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2 - \|\mathbf{w}''_t - \mathbf{u}_{t+1}\|_2^2) \\ &+ (\|\mathbf{w}''_t - \mathbf{u}_{t+1}\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2) = \delta_t + \psi_t + \epsilon_t + \rho_t. \end{aligned}$$

We have already proved the lower bounds for δ_t, ψ_t and ϵ_t above. For ρ_t, we have

$$\rho_t = 2\mathbf{v}_t \cdot \mathbf{w}''_t - 2\mathbf{v}_t \cdot \mathbf{u}_{t+1} - \|\mathbf{v}_t\|_2^2 .$$

It is obvious that $\mathbf{v}_t \cdot \mathbf{w}''_t = \|\mathbf{v}_t\|_2^2$, thus we have

$$\rho_t = \|\mathbf{v}_t\|_2^2 - 2\mathbf{v}_t \cdot \mathbf{u}_{t+1} \geq -2\mathbf{v}_t \cdot \mathbf{u}_{t+1}. \tag{3.19}$$

Applying the Cauchy-Schwarz inequality, we get

$$\rho_t \geq -2\|\mathbf{v}_t\|_2 \|\mathbf{u}_{t+1}\|_2 \geq -2\sqrt{\sigma}D. \tag{3.20}$$

Using Eqs. 3.4, 3.6 and 3.20, we obtain

$$\sum_{t=0}^{T} \Delta_t \geq m\left(\frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} - 2\sqrt{\sigma}D\right).$$

Since $\sum_t \Delta_t \leq D^2$, we have

$$m\left(\frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} - 2\sqrt{\sigma}D\right) \leq D^2.$$

Since $h = \frac{1-\alpha^2}{R^2} - 2(1+\alpha)\mu\beta - \frac{\alpha D^2}{2+\alpha} - 2\sqrt{\sigma}D > 0$, we obtain the result in the theorem. ∎

Similarly, for the algorithms with norm constraints, we obtain the following error bound:

THEOREM 3.4. *Assume that there exists an optimal sequence of weight vectors $\mathbf{u}_0, ..., \mathbf{u}_T \in \mathbb{R}^n$ such that $\|\mathbf{u}_t\|_2 = D \leq \beta$, $\ell_t^* = 0$ for all t, $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_2 \leq \mu$ and μ satisfies $q = \frac{1}{R^2} - 2\mu\beta - 2\sqrt{\sigma}D > 0$, then the number of errors made by the algorithm with the L2 norm constraint is bounded by $m \leq \frac{D^2}{q}$. Similarly, if there exists an optimal weight vector sequence such that $\|\mathbf{u}_t\|_1 = D$ and $\|\mathbf{u}_t - \mathbf{u}_{t+1}\|_\infty \leq \mu$, the number of errors made by the algorithm with the L1 norm constraint is bounded by $m \leq \frac{D^2}{q}$.*

Proof. Let $\Delta_t = \|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2$. We can prove the bound by lower and upper bounding $\sum_t \Delta_t$. As the above, we know $\sum_t \Delta_t \leq D^2$.

Obviously, $\Delta_t \neq 0$ only if $t \in I_T$. We will only consider this case here. Let $\mathbf{w}'_t$ be the solution of Problem 3.8. We define the vector $\mathbf{v}_t \in \mathbb{R}^n$ as

$$\mathbf{v}_{ti} = \begin{cases} \mathbf{w}'_{ti} & \text{if } \sum_{j=1}^{i} \mathbf{w}'^2_{tj} < \sigma \\ 0 & \text{otherwise.} \end{cases} \tag{3.21}$$

Then $\mathbf{w}_{t+1} = \mathbf{w}'_t - \mathbf{v}_t$. Δ_t can be rewritten as

$$(\|\mathbf{w}_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}'_t - \mathbf{u}_t\|_2^2) \tag{3.22}$$

$$+ (\|\mathbf{w}'_t - \mathbf{u}_t\|_2^2 - \|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2) \tag{3.23}$$

$$+ (\|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2). \tag{3.24}$$

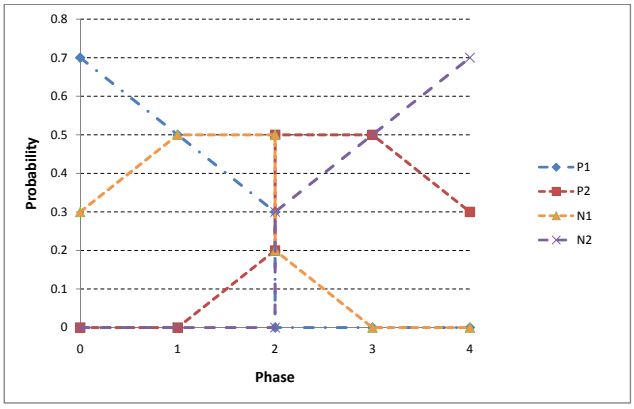


Figure 2: Non-stationary class probabilities employed in the experiments.

We have previously shown that $\chi_t + \gamma_t \geq \frac{1}{R^2} - 2\mu\beta$ and that $\|\mathbf{w}'_t - \mathbf{u}_{t+1}\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}_{t+1}\|_2^2 \geq -2\sqrt{\sigma}D$. Thus,

$$\Delta_t \geq \frac{1}{R^2} - 2\mu\beta - 2\sqrt{\sigma}D. \tag{3.25}$$

Since we know $\frac{1}{R^2} - 2\mu\beta - 2\sqrt{\sigma}D > 0$, we obtain the result in the theorem. ∎

This theorem shows that removing those small-weight features slightly increases the error bounds by subtracting $2\sqrt{\sigma}\|\mathbf{u}\|_2$ from the denominator. Given a small σ, ignoring the features with small weights has little influence on the prediction accuracy. Our experimental results show that the accuracy is not hurt even when we remove more than half of the features. This feature selection effect can also explain the superior performance of regularized online learning: our algorithms drive many feature weights towards zero. The algorithms tend to learn a model with a sparse parameter vector when the feature space is large.

4 Experimental Results

We evaluate our algorithms with three data sets.

4.1 Data sets. We tested our algorithms on one image data set, USPS, and two text data sets, 20NewsGroups and SRAA[1]. For all text documents, we removed the header and converted the remainder into a 0/1 vector of indicator variables for each word (without stemming or a stoplist). To make the experiments more realistic and dynamic, we defined a non-stationary process for generating the data sequence. In each data set, we chose four classes (P1, P2, N1, N2). We defined the union of P1 and P2 to be the positive class and the union of N1 and N2 to be the negative

[1] Available at http://www.cs.umass.edu/~mccallum/data/sraa.tar.gz

class. Then we defined four phases for the data sequence (see Figure 2). In Phase 1, the probability of class P1 decreases from 0.7 to 0.5 and the probability of N1 increases from 0.3 to 0.5. In Phase 2, the probability of P1 decreases from 0.5 to 0.3, P2 increases from 0 to 0.2, and N1 is fixed at 0.5. In Phase 3, P2 is fixed at 0.5, N1 decreases from 0.2 to 0, and N2 increases from 0.3 to 0.5. In Phase 4, P2 decreases from 0.5 to 0.3 and N2 increases from 0.5 to 0.7. Classes not mentioned have probability 0.

For the USPS set, we sampled 500 instances (without replacement) in each phase and defined P1 to be the digit "3", P2 to be "7", N1 to be "8", and N2 to be "9". For the 20NewsGroups and SRAA data sets, we sampled 800 instances (without replacement) in each phase. For 20NewsGroups, P1 is "Baseball", P2 is "Hockey", N1 is "Auto", and N2 is "Motorcycle". For SRAA, P1 is "RealAutos", P2 is "RealAviation", N1 is "SimulatedAuto", and N2 is "SimulatedAviation". There are 256 features in the USPS data, 8354 features in the 20NewsGroups data, and 9610 features in the SRAA data.

4.2 Approach. We applied our algorithms to the generated data sequence as follows. We used the first half of the data to select the regularization parameter and then trained the model on the first half. We then measured online performance using the second half of the data. Since the second half corresponds to phases 3 and 4, this approach means that the initial model (learned in phases 1 and 2) performs very badly initially, because of the discontinuous shift in class probabilities at the start of phase 3. The results are similar if the regularization parameters are tuned on the second half of the data. All results are averaged over 40 trials.

4.3 Comparison with Other Online Learning Methods. We compare our algorithms with the Bayesian Perceptron algorithm [32, 4] (which approximates the posterior with a Gaussian), the linear NORMA algorithm [21] (which is another regularized online learning algorithm), and the passive-aggressive algorithm [6] (which is a state-of-the-art online learning algorithm). The results are shown in Figure 1.

Of the five methods evaluated, our objective-regularized and L2-norm constrained algorithms are clearly superior on the USPS and 20NewsGroups data sets. On SRAA, our algorithms are able to adapt very rapidly to the nonstationary change in class probabilities at the start of Phase 3. However, by the end of Phase 4, the Bayesian Perceptron algorithm has caught up and surpassed our methods.

We consider each of the methods in more detail. The worst method appears to be the NORMA algorithm. Although in theory, NORMA can quickly adjust its hypothesis, in our experiments, it adjusts more slowly than all of the other algorithms. NORMA is consistently the worst performer, although PA is tied with it on SRAA during the early part of Phase 3.

The Passive-Aggressive algorithm is the next worse performer. On USPS and 20NewsGroups, it starts out adapting more rapidly than NORMA and the Bayesian Perceptron, but then its performance fails to improve as rapidly as the other methods.

The Bayesian Perceptron algorithm initially adapts more slowly to the non-stationary change at the start of Phase 3 than our algorithms. However, by the end of Phase 4, its accuracy approaches that of our algorithms, and in the case of SRAA, it eventually becomes more accurate than our methods. This may show that our algorithms are more strongly designed for non-stationary distributions and suffer somewhat when the distribution is fairly stable for a sufficiently long time. The Bayesian Perceptron algorithm requires more CPU time and memory, partly because it must maintain a covariance matrix. When the feature space is extremely large, the Bayesian Perceptron algorithm will become infeasible.

The experiments show that both the L1-norm constrained algorithm and the L2-norm constrained algorithm give similar performance. A possible reason for this is that the number of irrelevant features is within an order of magnitude of the number of informative features in these problems. As we will show below, more than half of the features are informative in the USPS problem. For the text problems, there are several times more meaningless features than informative features, but the number of informative features is still quite large. Previous work suggests that L1 regularization works best when a small number of features play moderate-sized effects in classification, while L2 regularization is quite competitive when a large number of features play small effects in classification [33]. One future direction would be to consider the elastic net approach, which employs a mix of L1 and L2 regularization [36].

One advantage of the L2-constrained method is that it is more efficient than the L1-constrained method. However, recent work has shown that L1-constrained methods can be made very fast [36, 11]. In our implementation, we used the current weight vector as the initial value when solving the quadratic programming problem to compute the update. This makes the update quite efficient. For example, it took less than 10 minutes to process all 3200 20NewsGroups instances on a Linux machine (AMD64 2.6Ghz CPU, 2G memory). As a comparison, the objective-regularized algorithm took about 16 seconds to process all 3200 20NewsGroups instances.

Feature Selection Power. Next we investigate the feature selection power of our algorithms and compare this to two other feature selection methods. For our methods, we did the following. We chose a set of possible values for σ and ran each algorithm with each value. In each run at each time step, we sorted the weights by their magnitudes and removed features until the sum of the absolute values of

the weights exceeded σ. We computed the average number of non-zero weights per iteration and the overall error rate over the testing sequence. Features were "removed" only for purposes of predicting the label on the new example. The algorithm continued using the entire weight vector to perform its weight updates.

The other two feature selection methods that we explored were information gain [35] and a game-theoretic method. When updating the model, we first perform feature selection and then carry out the model update. Information gain is a pair-wise feature selection method that is very effective in aggressive term removal without losing classification accuracy in stationary environments [35]. It simply computes the mutual information between each feature and the class label: $I(x_j; y)$. It then chooses the k features with highest mutual information.

We also devised a game-theoretic feature selection algorithm that tries to maximize accuracy in the worst case. Online learning faces a dynamic, sometimes even adversarial environment [14, 9]. We treat feature selection as a two-person game [12]. Let the learner be the row player and the environment be the column player. The game can be thought as this: given an observation $\mathbf{x}$, the row player chooses a feature x_i and probabilistically determines $y[i]$, the label of $\mathbf{x}$ based only on feature x_i. Simultaneously, the column player chooses a feature x_j and probabilistically determines the label $y[j]$. If they predict the same label, then the row player gets reward 1, otherwise 0. We want to design a reward matrix M so that each element $M(i, j)$ is proportional to the likelihood that $y[i] = y[j]$ when the row player selects feature x_i and the column player selects feature x_j. There could be many choices. We chose the mutual information between x_i and x_j: $M[i, j] := I(x_i; x_j)$. Let $\mathbf{u}$ be the learner's selection strategy and $\mathbf{v}$ be the environment's strategy, our goal is then to find a strategy that can maximize the reward in the worst case:

$$\max_{\mathbf{u}} \min_{\mathbf{v}} (\mathbf{u} M \mathbf{v}) \tag{4.26}$$
$$\text{subject to } \forall i \;\; u_i \in \{0, 1\}, \quad \forall j \;\; v_j \in \{0, 1\},$$
$$\sum_i u_i = \sum_j v_j = k,$$

where k is the number of selected features. We then select those features with $u_i = 1$. In practice, we can relax the integer requirement and replace the constraint $\forall i \;\; u_i \in \{0, 1\}, \;\; \forall j \;\; v_j \in \{0, 1\}$ with constraint $\forall i \; 0 \leq u_i \leq 1, \quad \forall j \quad 0 \leq v_j \leq 1$. The problem can be converted into a linear programming problem and solved efficiently. We then select the k features with the largest u_i.

The feature selection results are plotted in Figure 3. In most cases, the regularized methods significantly outperform other feature selection methods, especially for the USPS problem in which most features are informative. Information gain does not realize that the data is changing and continues to select features aggressively based on overall mutual information. Its performance is extremely bad when we select only a small number of features. Information gain shows improvement only for the 20NewsGroups problem, which is relatively easy to predict.

On USPS, the game-theoretic method selects better feature sets than information gain. However, the game-theoretic method is too inefficient to run on 20NewsGroups or SRAA because of the large number of features in these domains.

The L1-norm constrained method does not perform as well as the L2-norm method. This is probably because the L1-norm method is more likely to put more weight on a smaller number of informative features. Once these features are removed, its performance is severely damaged. Although it shows slightly better performance when we allow many features, it becomes unstable and its accuracy is much worse than the L2 constrained method when the number of selected features is small. The accuracy of the L2-constrained method is not hurt even when we remove most of the features. This suggests that this method has the ability to learn a sparse model that still performs well.

4.4 Adapting to a Changing Environment. For online learning, the learned model is determined by the *active set*. The regularized methods naturally reduce the influence of the old active instances and pay more attention to the more recent ones. In Figure 4, we compare the final learned models for the USPS problem. We plot the learned models as follows: each feature corresponds to its position in the image, and the gray scale is proportional to its absolute weight. To compute the information gain figure, we apply the information gain criterion to the entire data sequence and compute the information gain of each feature. For the Passive-Aggressive (PA) algorithm, we plot the absolute weights of the features at the end of Phase 4. At the end of Phase 4, each algorithm should be just predicting whether a digit is "7" or "9". The images suggest that this is what the regularized algorithms (and to a lesser extent, PA) are doing. The information gain method does not appear to be focused on this task. We can also see that the L1-constrained method has more aggressively put larger weights on a smaller number of features. The PA algorithm still puts large weights on some features that can discriminate "3" and "8" (from Phases 1 and 2).

5 Application to Activity Recognition

We applied a variation of our regularized online learning algorithms to do activity recognition in an intelligent activity management system, the TaskTracer system [10, 30, 29].

Figure 5: The screenshot of Task Selector, an application that is located at the right bottom of the desktop.

5.1 The TaskTracer System and Its Activity Recognition Problem The TaskTracer system is an intelligent activity management system that helps knowledge workers manage their work based on two assumptions: (a) the user's work can be organized as a set of ongoing activities such as "Write TaskTracer Paper" or "Take CS534 Class", (b) each activity is associated with a set of *resources*. "Resource" is an umbrella term for documents, folders, email messages, email contacts, web pages and so on. TaskTracer collects events generated by the user's computer-visible behavior, including events from MS Office, Internet Explorer, Windows Explorer, and the Windows XP operating system. The user can declare a "current activity" (via a "Task Selector" user interface, see Figure 5), and TaskTracer records all resources as they are accessed and associates them with the current declared activity. TaskTracer then configures the desktop in various ways to support the user. For example, it supplies an application called the Task Explorer, which presents a unified view of all of the resources associated with the current activity and makes it easy to open those resources in the appropriate application. It also predicts which folder the user is most likely to access and modifies the Open/Save dialog box to use that folder as the default folder. If desired, it can restore the desktop to the state it was in when the user last worked on an activity, and so on.

To get the right resource-activity associations, the user must remember to indicate the current declared activity each time the activity changes. If the user forgets to do this, then resources become associated with incorrect activities, and TaskTracer becomes less useful. For this reason, we sought to supplement the user's manual activity declaration with an activity predictor that attempts to predict the current activity of the user. If the predictor is sufficiently accurate, its predictions could be applied to associate resources with activities, to propose correct folders for files and email, and to remind the user to update the current declared activity.

We have previously developed two machine learning methods for detecting activity switches and predicting the current activity of the user [31, 30]. A significant drawback of both of these learning methods is that they employed a batch SVM algorithm [19, 5]. This must store all of the training examples and reprocess all of them when retraining is required. SVM training time is roughly quadratic in the number of examples, so the longer TaskTracer is used, the slower the training becomes. Furthermore, activity prediction is a multi-class prediction problem over a potentially large number of classes. For example, our busiest user worked on 299 different activities during a four-month period. To perform multi-class learning with SVMs, we employed the one-versus-rest approach. If there are K classes, then this requires learning K classifiers. If there are N examples per class, then the total running time is roughly $O(N^2K^3)$. This is not practical in an interactive, desktop setting.

Another drawback with the previous learning methods is that they are slow to respond to the user's feedback. In many cases, when the user provides feedback, the algorithm only takes a small step in the right direction, so the user must repeatedly provide feedback until enough steps have been taken to fix the error. This behavior has been reported to be extremely annoying.

Our regularized online learning methods can address the above problems. Thus we redesigned the activity predictor to apply our new methods.

5.2 System Design As indicated above, TaskTracer monitors various desktop events (Open, Close, Save, SaveAs, Change Window Focus and so on). Every s seconds (default s=60) or when the user declares an activity switch, it computes an information vector $\mathbf{X}_t$ describing the time interval t since the last information vector was computed. This information vector is then mapped into feature vectors by two functions: $\mathbf{F}_A : (\mathbf{X}_t, y_j) \rightarrow R^k$ and $\mathbf{F}_S : (\mathbf{X}_t) \rightarrow R^m$. The first function $\mathbf{F}_A$ computes *activity-specific* features for a specified activity y_j; the second function $\mathbf{F}_S$ computes *switch-specific features*. The activity-specific features include

- Strength of association of the active resource with activity y_j: if the user has explicitly declared that the active resource belongs to y_j (e.g., by drag-and-drop in TaskExplorer), the current activity is likely to be y_j. If the active resource was implicitly associated with y_j for some duration (which happens when y_j is the declared activity and then the resource is visited), this is a weaker indication that the current activity is y_j.

- Percentage of open resources associated with activity y_j: if most open resources are associated with y_j, it is likely that y_j is the current activity.

- Importance of window title word x to activity y_j. Given the bag of words Ω from the window title, we compute a variant of TF-IDF [6] for each word x and activity y_j: $\text{TF}(x, \Omega) \cdot \log \frac{|\overline{S}|}{\text{DF}(x, \overline{S})}$. Here, $\overline{S}$ is the set of all feature vectors not labeled as y_j, $\text{TF}(x, \Omega)$ is the number of times x appears in Ω and $\text{DF}(x, \overline{S})$ is the number of feature vectors containing x that are not labeled y_j.

These activity-specific features are intended to predict whether y_j is the current activity. The switch-specific features predict the likelihood of a switch. They include

- Number of resources closed in the last s seconds: if the user is switching activities, many open resources will often be closed.
- Percentage of open resources that have been accessed in the last s seconds: if the user is still actively accessing open resources, it is unlikely there is an activity switch.
- The time since the user's last explicit activity switch: immediately after an explicit switch, it is unlikely the user will switch again. But as time passes, the likelihood of an undeclared switch increases.

To detect an activity switch, we adopt a sliding window approach: at time t, we use two information vectors ($\mathbf{X}_{t-1}$ and $\mathbf{X}_t$) to score every pair of activities for time intervals $t-1$ and t. Given an activity pair $\langle y_{t-1}, y_t \rangle$, the scoring function g is defined as

$$g(\langle y_{t-1}, y_t \rangle) = \Lambda_1 \cdot \mathbf{F}_A(\mathbf{X}_{t-1}, y_{t-1}) + \Lambda_1 \cdot \mathbf{F}_A(\mathbf{X}_t, y_t) - I[y_{t-1} \neq y_t] \left(\Lambda_2 \cdot \mathbf{F}_S(\mathbf{X}_{t-1}) + \Lambda_3 \cdot \mathbf{F}_S(\mathbf{X}_t)\right),$$

where $\Lambda = \langle \Lambda_1, \Lambda_2, \Lambda_3 \rangle \in \mathbb{R}^n$ is a set of weights to be learned by the system, $I[p] = 1$ if p is true and 0 otherwise, and the dot $(\cdot)$ means inner product. The first two terms of g measure the likelihood that y_{t-1} and y_t are the activities at time $t-1$ and t (respectively). The third term measures the likelihood that there is no activity switch from time $t-1$ to t. Thus, the third component of g serves as a "switch penalty" when $y_{t-1} \neq y_t$.

We search for the $\langle \hat{y}_1, \hat{y}_2 \rangle$ that maximizes the score function g. If $\hat{y}_2$ is different from the current declared activity and the score is larger than a specified threshold, then a switch prediction is made.

5.3 Efficient Learning Algorithm. We employ a soft-margin variant of the L2-regularized objective algorithm. All of the algorithms previously discussed in this paper are "hard margin" algorithms that ensure that each new example is correctly classified with a functional margin of 1. This is appropriate for non-stationary learning problems, but it may not be a good approach when there are likely to be errors in the training data. Specifically, in the TaskTracer setting, the user can easily mislabel an activity switch. Hence, our algorithm needs to be robust to such label errors.

To achieve this, we follow the standard soft-margin approach by introducing a slack variable ξ in the constraint and penalizing it in the objective function:

$$\Lambda_{t+1} = \arg\min_{\Lambda \in \mathbb{R}^n} \frac{1}{2} \|\Lambda - \Lambda_t\|_2^2 + C\xi^2 + \frac{\alpha}{2} \|\Lambda\|_2^2$$

$$\text{subject to } g(\langle y_1, y_2 \rangle) - g(\langle \hat{y}_1, \hat{y}_2 \rangle) \geq 1 - \xi. \tag{5.27}$$

The second term in the objective function, $C\xi^2$, serves to encourage ξ to be small. Ideally, $\xi = 0$, so that this enforces the condition that the functional margin (between correct and incorrect scores) should be 1. The constant parameter C controls the tradeoff between taking small steps (the first term) and fitting the data (driving ξ to zero).

Like our other algorithms, this has a closed-form solution, so it can be computed in time linear in the number of features and the number of classes. We now derive the closed-form update rule for this modified version of our algorithms. First, let us expand the g terms. Define $\mathbf{Z}_t = \langle \mathbf{Z}_t^1, \mathbf{Z}_t^2, \mathbf{Z}_t^3 \rangle$ where

$$\begin{aligned}
\mathbf{Z}_t^1 =& \mathbf{F}_A(\mathbf{X}_{t-1}, y_1) + \mathbf{F}_A(\mathbf{X}_t, y_2) \\
& - \mathbf{F}_A(\mathbf{X}_{t-1}, \hat{y}_1) - \mathbf{F}_A(\mathbf{X}_t, \hat{y}_2) \\
\mathbf{Z}_t^2 =& \left(I[\hat{y}_1 \neq \hat{y}_2] - I[y_1 \neq y_2]\right) \mathbf{F}_S(\mathbf{X}_{t-1}) \\
\mathbf{Z}_t^3 =& \left(I[\hat{y}_1 \neq \hat{y}_2] - I[y_1 \neq y_2]\right) \mathbf{F}_S(\mathbf{X}_t).
\end{aligned}$$

Plugging this into the above optimization problem allows us to rewrite the inequality constraint as the following simple form:

$$\Lambda \cdot \mathbf{Z}_t \geq 1 - \xi.$$

We can then derive the following result:

LEMMA 5.1. *The optimization problem (5.27) has the closed-form solution*

$$\Lambda_{t+1} := \frac{1}{1+\alpha}(\Lambda_t + \tau_t \mathbf{Z}_t), \tag{5.28}$$

where

$$\tau_t = \frac{1 - \Lambda_t \cdot \mathbf{Z}_t + \alpha}{\|\mathbf{Z}_t\|_2^2 + \frac{1+\alpha}{2C}}. \tag{5.29}$$

A detailed proof is presented in the Appendix. The update rule (5.28) can be viewed as shrinking the current weight vector Λ_T and then adding in the incorrectly-classified training example with a step size of $\tau_t/(1+\alpha)$. The step size is determined (in the numerator of (5.29)) by the size of the error $(1 - \Lambda_t \cdot \mathbf{Z}_t)$ and (in the denominator) by the squared length of the feature vector and a correction term involving α and C.

The time to compute this update is linear in the number of features. Furthermore, the cost does not increase with the number of classes, because the update involves comparing only the predicted and correct classes.

It is worth asking how much accuracy is lost in order to obtain such an efficient online algorithm compared to a batch algorithm (or, more generally, to the best possible algorithm). By extending existing results from computational learning theory, we can provide a partial answer to this question. Let us assume that the length of each vector $\mathbf{Z}_t$ is no more than some fixed value R:

$$\|\mathbf{Z}_t\|_2 \leq R.$$

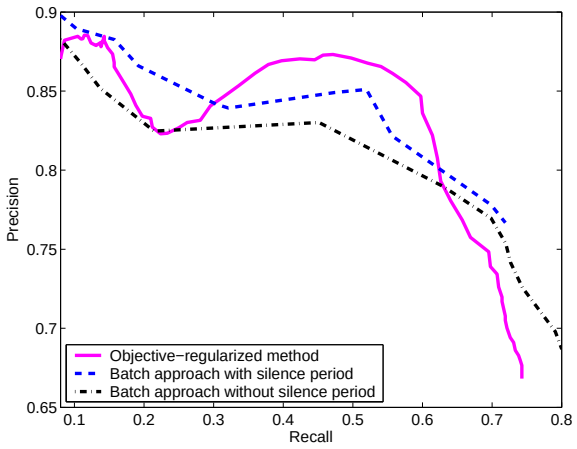


Figure 6: Precision of different learning methods as a function of the recall, created by varying the alarm threshold.

This is easily satisfied if every feature has a fixed range of values. Suppose there is some other algorithm that computes a better weight vector $\mathbf{u} \in \mathbb{R}^n$. Let $\ell_t^* = \max\{0, 1 - \mathbf{u} \cdot \mathbf{Z}_t\}$ denote the hinge loss of $\mathbf{u}$ at iteration t. With these definitions, we can obtain a result that compares the accuracy of the online algorithm after T examples to the total hinge loss of the weight vector $\mathbf{u}$ after T examples:

THEOREM 5.1. *Assume that there exists a vector* $\mathbf{u} \in \mathbb{R}^n$ *which satisfies* $h = \frac{1-\alpha^2}{R^2 + \frac{1+\alpha}{2C}} - \frac{\alpha}{2+\alpha} \|\mathbf{u}\|_2^2 > 0$. *Given* $\alpha < 1$, *the number of prediction mistakes made by our algorithm is bounded by* $m \leq \frac{1}{h} \|\mathbf{u}\|_2^2 + \frac{2C}{h(1+\alpha)} \sum_{t=0}^{T} (\ell_t^*)^2$.

A detailed proof is presented in the Appendix. This theorem tells us that the number of mistakes made by our online algorithm is bounded by the sum of (a) the squared hinge loss of the ideal weight vector $\mathbf{u}$ multiplied by a constant term and (b) the squared Euclidean norm of $\mathbf{u}$ divided by h. This bound is not particularly tight, but it does suggest that as long as $\mathbf{u}$ is not too large, the online algorithm will only make a constant factor more errors than the ideal (batch) weight vector.

Theoretical analysis can provide one more insight into the relationship between C and how often the user makes mistaken feedbacks. Since usually $R^2 \gg \frac{1+\alpha}{2C}$, we can treat h as approximately independent of C. When the user makes many feedback errors (i.e., $\sum_{t=0}^{T} (\ell_t^*)^2$ is large), we should set C to be small to allow a large slack value ξ. As C becomes smaller, $\frac{2C}{h(1+\alpha)} \sum_{t=0}^{T} (\ell_t^*)^2$ becomes smaller, too. Thus we will have a smaller guaranteed error bound.

5.4 Experimental Results on Real Desktop User Data. We deployed TaskTracer on Windows machines in our research group and collected data from one regular user who was fairly careful about declaring switches. This data set records 4 months of daily work, which involved 299 distinct activities, 65,049 instances (i.e., information vectors), and 3,657 activity switches.

To evaluate the learning algorithm described in Section 5.3, we make the assumption that these activity switches are all correct, and we perform the following simulation. The data is ordered according to time. Suppose the user forgets to declare every fourth switch. We feed the instances to the online algorithm and ask it to make predictions. A switch prediction is treated as correct if the predicted activity is correct and if the predicted time of the switch is within 5 minutes of the real switch point. When a prediction is made, our simulation provides the correct time and activity as feedback. When the user declares an activity switch, the simulation also generates training instances [29].

The online algorithm parameters were set based on experiments with non-TaskTracer benchmark sets. We set $C = 10$ (which is a value widely used in the literature) and $\alpha = 0.001$ (which gave good results on the benchmark data sets).

Performance is measured by *precision* and *recall*. Precision is the number of switches correctly predicted divided by the total number of switch predictions, and Recall is the number of switches correctly predicted divided by the total number of undeclared switches. We obtain different precision and recall values by varying the score confidence threshold required to make a prediction.

The results comparing our online learning approach with the batch training methods are plotted in Figures 6. The batch training methods adopt a large margin approach [34] to train the weight vector Λ. This method creates a positive example for the correct label and a negative example the the incorrect label. It then trains the weight vector using all available examples with the linear kernel [5]. To reduce repeated incorrect predictions, we also evaluated the batch training method configured so that it does not raise any new alarms for at least the first 10 minutes after the most recent switch alarm. Our online approach gives competitive performances compared to the batch approaches. The non-silent batch approach only takes a small step in the right direction after the user provides feedback. So it is likely that it will make the same mistake in the next iteration. The user has to repeatedly provide feedback so that enough steps are taken to fix the error. Our online learning approach outperforms this in most cases. If we force the batch predictor to keep silent after a switch alarm, the batch predictor can avoid many repeated mistakes. In this configuration during the silent period, the batch predictor treats repeated prediction as an error. Hence, by the time the silent period is over, it has probably fixed the error by gaining enough training examples.

Compared to the batch approaches, the online learning

approach is much more efficient. On an ordinary PC, it only took 4 minutes to make predictions for the 65,049 instances (0.0037 seconds/instance) while the non-silent batch approach needed more than 37 hours (more than 2 seconds/instance), a speedup factor of more than 500 fold.

6 Conclusions And Further Work

An important challenge for online learning is to achieve high accuracy while also adapting rapidly to changes in the environment. This paper presented three efficient online algorithms that combine large margin training with regularization methods that enable rapid adaptation to nonstationary environments. Each algorithm comes with theoretical guarantees in the form of online error bounds with respect to an optimal online algorithm. The algorithms have some interesting characteristics that make them especially appropriate in dynamic environments. First, they shrink the weights towards zero, which makes it easier to adapt the weights when the environment suddenly changes. Second, the algorithms naturally shrink the influence of the old instances and put more weight on the more recent ones. Third, the methods learn a sparse model that ignores or down-weights irrelevant features. Finally, we have successfully applied a soft-margin version of our algorithms to activity recognition in the TaskTracer system [29].

There are two promising directions for future work. First, we noted that the learned weight vector of our L2-regularized methods is a linear combination of the active instances. It would be interesting to employ the kernel trick here by replacing the inner product with a kernel satisfying the Mercer conditions and compare our algorithms with the Forgetron algorithm [8]. The Forgetron is an online kernel-based learning algorithm that controls the number of support vectors by removing the oldest support vector when the number of support vectors exceeds a fixed quota. Since removing a support vector may significantly change the hypothesis, the Forgetron aggressively "shrinks" the weight of old support vectors. Our L2-regularized online algorithms naturally shrink the weights of those support vectors, and we could also control the number of support vectors by removing the oldest one.

Our current algorithms assume that the system will always receive the correct label immediately after making a prediction. However, in many cases there can be a delay between the time a prediction is made and the time feedback is received. For example, in TaskTracer, sometimes there is no feedback at all after the system makes a switch alarm, because the user is too busy and does not notice the alarm. Currently TaskTracer does nothing when this happens, and the system assumes that its prediction was incorrect. But an alternative would be to assume the prediction is correct and automatically change TaskTracer's "current activity". This situation can be thought as a bandit setting [1, 20]. It would be very interesting to design a bandit version of our algorithms.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments and suggestions.

Appendix – Detailed Proof of the Theorems

Proof of Lemma 3.2

Proof. The Lagrangian of Problem 3.8 is

$$L(\mathbf{w},\tau) = \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|_2^2 + \tau(1-y_t(\mathbf{w}\cdot\mathbf{x}_t)) + \lambda(\|\mathbf{w}\|_2^2-\beta^2),$$

where $\tau \geq 0$ and $\lambda \geq 0$ are the Lagrange multipliers. Differentiating this Lagrangian with respect to the elements of $\mathbf{w}$ and setting the partial derivative to zero gives

$$\mathbf{w} = \frac{1}{1+2\lambda}(\mathbf{w}_t + \tau y_t \mathbf{x}_t). \tag{6.30}$$

We let $Z = 1+2\lambda$. The KKT conditions require constraint $1-y_t(\mathbf{w}\cdot\mathbf{x}_t) \leq 0$ to be active, which leads to

$$\tau = \frac{Z - y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{\|\mathbf{x}_t\|_2^2}. \tag{6.31}$$

The KKT conditions require $\lambda(\|\mathbf{w}\|_2^2-\beta^2)=0$. We discuss two cases here. First, if $\lambda = 0$, then we get $\tau = \frac{1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{\|\mathbf{x}_t\|_2^2}$ from Equation 6.31.

Second, if $\|\mathbf{w}\|_2^2 - \beta^2 = 0$. We replace τ with $\frac{Z-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{\|\mathbf{x}_t\|_2^2}$ and get

$$\|\mathbf{w}_t\|_2^2 + \frac{2Zy_t(\mathbf{w}_t\cdot\mathbf{x}_t) - 2(\mathbf{w}_t\cdot\mathbf{x}_t)^2}{\|\mathbf{x}_t\|_2^2} + \frac{Z^2 - 2Zy_t(\mathbf{w}_t\cdot\mathbf{x}_t) + (\mathbf{w}_t\cdot\mathbf{x}_t)^2}{\|\mathbf{x}_t\|_2^2} = \beta^2 Z^2$$
$$(\beta^2\|\mathbf{x}_t\|_2^2 - 1)Z^2 = \|\mathbf{w}_t\|_2^2\|\mathbf{x}_t\|_2^2 - (\mathbf{w}_t\cdot\mathbf{x}_t)^2.$$

This can be possible only if $\beta^2\|\mathbf{x}_t\|_2^2 - 1 \leq \|\mathbf{w}_t\|_2^2\|\mathbf{x}_t\|_2^2 - (\mathbf{w}_t\cdot\mathbf{x}_t)^2$ since $Z = 1+2\lambda \geq 1$. In this case, we get $Z = \sqrt{\frac{\|\mathbf{w}_t\|_2^2\|\mathbf{x}_t\|_2^2 - (\mathbf{w}_t\cdot\mathbf{x}_t)^2}{\beta^2\|\mathbf{x}_t\|_2^2 - 1}}$. If $\beta^2\|\mathbf{x}_t\|_2^2 - 1 > \|\mathbf{w}_t\|_2^2\|\mathbf{x}_t\|_2^2 - (\mathbf{w}_t\cdot\mathbf{x}_t)^2$, this corresponds to the first case, $\lambda = 0$. We can easily show that constraint $\|\mathbf{w}\|_2^2 \leq \beta^2$ is always feasible and inactive if it is true:

$$\|\mathbf{w}\|_2^2 = \frac{\|\mathbf{w}_t\|_2^2\|\mathbf{x}_t\|_2^2 - (\mathbf{w}_t\cdot\mathbf{x}_t)^2 + 1}{\|\mathbf{x}_t\|_2^2} < \frac{\beta^2\|\mathbf{x}_t\|_2^2 - 1 + 1}{\|\mathbf{x}_t\|_2^2} = \beta^2.$$

Since both our objective function and constraint functions are convex, we know our solution is globally optimal. Combining the above two cases, we conclude the proof. ∎

Proof of Lemma 5.1

Proof. The Lagrangian of the optimization problem in (3.1) is

$$L(\Lambda, \xi, \tau) = \frac{1}{2}\left\|\Lambda - \Lambda_t\right\|_2^2 + C\xi^2 + \frac{\alpha}{2}\left\|\Lambda\right\|_2^2 + \tau(1 - \Lambda \cdot \mathbf{Z}_t - \xi), \tag{6.32}$$

where $\tau \geq 0$ is the Lagrange multiplier. Differentiating this Lagrangian with respect to the elements of Λ and setting the partial derivative to zero gives

$$\Lambda = \frac{1}{1+\alpha}\Lambda_t + \frac{\tau}{1+\alpha}\mathbf{Z}_t. \tag{6.33}$$

Differentiating the Lagrangian with respect to ξ and setting the partial derivative to zero gives

$$\xi = \frac{\tau}{2C}. \tag{6.34}$$

Expressing ξ as above and replacing Λ in Eq 6.32 with Eq 6.33, the Lagrangian becomes

$$\begin{aligned} L(\tau) =& \frac{1}{2}\left\|\frac{\tau}{1+\alpha}\mathbf{Z}_t - \frac{\alpha}{1+\alpha}\Lambda_t\right\|_2^2 + \frac{\tau^2}{4C} \\ &+ \frac{\alpha}{2}\left\|\frac{\tau}{1+\alpha}\mathbf{Z}_t + \frac{1}{1+\alpha}\Lambda_t\right\|_2^2 \\ &+ \tau\left(1 - \frac{\Lambda_t \cdot \mathbf{Z}_t}{1+\alpha} - \frac{\tau\left\|\mathbf{Z}_t\right\|_2^2}{1+\alpha} - \frac{\tau}{2C}\right). \end{aligned}$$

By setting the derivative of the above to zero, we get

$$1 - \frac{\tau}{2C} - \frac{\tau}{1+\alpha}\left\|\mathbf{Z}_t\right\|_2^2 - \frac{(\Lambda_t \cdot \mathbf{Z}_t)}{1+\alpha} = 0 \tag{6.35}$$

$$\Rightarrow \quad \tau = \frac{1 - (\Lambda_t \cdot \mathbf{Z}_t) + \alpha}{\left\|\mathbf{Z}_t\right\|_2^2 + \frac{1+\alpha}{2C}}. \tag{6.36}$$

Combining Eq 6.33 and Eq 6.36 completes the proof. ∎

Proof of Theorem 5.1

Proof. Let $\Delta_t = \left\|\Lambda_t - \mathbf{u}\right\|_2^2 - \left\|\Lambda_{t+1} - \mathbf{u}\right\|_2^2$. We can prove the bound by lower and upper bounding $\sum_t \Delta_t$.

Since Λ_0 is a zero vector and the norm is always non-negative, $\sum_t \Delta_t = \left\|\Lambda_0 - \mathbf{u}\right\|_2^2 - \left\|\Lambda_T - \mathbf{u}\right\|_2^2 \leq \left\|\Lambda_0 - \mathbf{u}\right\|_2^2 = \left\|\mathbf{u}\right\|_2^2$.

Obviously, $\Delta_t \neq 0$ only if $t \in I_T$. We will only consider this case here. Let $\Lambda'_t = \Lambda_t + \tau_t\mathbf{Z}_t$, $\Lambda_{t+1} = \frac{1}{1+\alpha}\Lambda'_t$. Δ_t can be rewritten as

$$\begin{aligned} &(\left\|\Lambda_t - \mathbf{u}\right\|_2^2 - \left\|\Lambda'_t - \mathbf{u}\right\|_2^2) + (\left\|\Lambda'_t - \mathbf{u}\right\|_2^2 - \left\|\Lambda_{t+1} - \mathbf{u}\right\|_2^2) \\ &= \delta_t + \epsilon_t. \end{aligned} \tag{6.37}$$

We will lower bound both δ_t and ϵ_t. Let $\psi^2 = (1+\alpha)/2C$. For δ_t, we have

$$\begin{aligned} \delta_t &= \left\|\Lambda_t - \mathbf{u}\right\|_2^2 - \left\|\Lambda_t + \tau_t\mathbf{Z}_t - \mathbf{u}\right\|_2^2 && (6.38) \\ &= \left\|\Lambda_t - \mathbf{u}\right\|_2^2 - \left\|\Lambda_t - \mathbf{u}\right\|_2^2 \\ &\quad - 2\tau_t\mathbf{Z}_t \cdot (\Lambda_t - \mathbf{u}) - \left\|\tau_t\mathbf{Z}_t\right\|_2^2 && (6.39) \\ &\geq 2\tau_t\ell_t - 2\tau_t\ell_t^* - \tau_t^2\left\|\mathbf{Z}_t\right\|_2^2 && (6.40) \\ &\geq 2\tau_t\ell_t - 2\tau_t\ell_t^* - \tau_t^2\left\|\mathbf{Z}_t\right\|_2^2 - (\psi\tau_t - \ell_t^*/\psi)^2 && (6.41) \\ &= 2\tau_t\ell_t - \tau_t^2(\left\|\mathbf{Z}_t\right\|_2^2 + \psi^2) - (\ell_t^*)^2/\psi^2. && (6.42) \end{aligned}$$

We get Eq 6.41 because $(\psi\tau_t - \ell_t^*/\psi)^2 \geq 0$. Plugging the definition of τ_t and considering $\ell_t \geq 1$, we get

$$\begin{aligned} \delta_t &\geq \frac{\ell_t^2 - \alpha^2}{\left\|\mathbf{Z}_t\right\|_2^2 + \frac{1+\alpha}{2C}} - \frac{2C}{1+\alpha}(\ell_t^*)^2 && (6.43) \\ &\geq \frac{1-\alpha^2}{R^2 + \frac{1+\alpha}{2C}} - \frac{2C}{1+\alpha}(\ell_t^*)^2. && (6.44) \end{aligned}$$

For ϵ_t, we have

$$\epsilon_t = (1 - \frac{1}{(1+\alpha)^2})\left\|\Lambda'_t\right\|_2^2 - 2(1 - \frac{1}{1+\alpha})\Lambda'_t \cdot \mathbf{u}. \tag{6.45}$$

Using the fact that $\left\|\mathbf{u} - \mathbf{v}\right\|_2^2 \geq 0$ which equals to $\left\|\mathbf{u}\right\|_2^2 - 2\mathbf{u} \cdot \mathbf{v} \geq -\left\|\mathbf{v}\right\|_2^2$, we get

$$\begin{aligned} &(1 - \frac{1}{(1+\alpha)^2})\left\|\Lambda'_t\right\|_2^2 - 2(1 - \frac{1}{1+\alpha})\Lambda'_t \cdot \mathbf{u} && (6.46) \\ &\geq -\frac{1 - \frac{1}{1+\alpha}}{1 + \frac{1}{1+\alpha}}\left\|\mathbf{u}\right\|_2^2 = -\frac{\alpha}{2+\alpha}\left\|\mathbf{u}\right\|_2^2. && (6.47) \end{aligned}$$

Using Eq 6.44 and 6.47, we get

$$\begin{aligned} &\sum_{t=0}^{T}\Delta_t = \sum_{t \in I_T}\Delta_t && (6.48) \\ &\geq \sum_{t \in I_T}\left((\frac{1-\alpha^2}{R^2 + \frac{1+\alpha}{2C}} - \frac{2C}{1+\alpha}(\ell_t^*)^2) - \frac{\alpha}{2+\alpha}\left\|\mathbf{u}\right\|_2^2\right) && (6.49) \\ &= m(\frac{1-\alpha^2}{R^2 + \frac{1+\alpha}{2C}} - \frac{\alpha}{2+\alpha}\left\|\mathbf{u}\right\|_2^2) - \sum_{t \in I_t}\frac{2C}{1+\alpha}(\ell_t^*)^2 && (6.50) \\ &\geq m(\frac{1-\alpha^2}{R^2 + \frac{1+\alpha}{2C}} - \frac{\alpha}{2+\alpha}\left\|\mathbf{u}\right\|_2^2) - \frac{2C}{1+\alpha}\sum_{t=0}^{T}(\ell_t^*)^2. && (6.51) \end{aligned}$$

Since $\sum_t \Delta_t \leq \left\|\mathbf{u}\right\|_2^2$, we have

(6.52)

$$m\left(\frac{1-\alpha^2}{R^2+\frac{1+\alpha}{2C}}-\frac{\alpha}{2+\alpha}\left\|\mathbf{u}\right\|_2^2\right)-\frac{2C}{1+\alpha}\sum_{t=0}^{T}(\ell_t^*)^2\leq\left\|\mathbf{u}\right\|_2^2.$$

Since $h=\frac{1-\alpha^2}{R^2+\frac{1+\alpha}{2C}}-\frac{\alpha}{2+\alpha}\left\|\mathbf{u}\right\|_2^2>0$, we get the result in the theorem. ∎

References

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. of FOCS-95*, pages 322–331, 1995.

[2] V. R. Carvalho and W. W. Cohen. Single-pass online learning: performance, voting schemes and online feature selection. In *Proc. of KDD-06*, pages 548–553, 2006.

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

[4] K. M. A. Chai, H. L. Chieu, and H. T. Ng. Bayesian online classifiers for text classification and filtering. In *Proc. of SIGIR-02*, pages 97–104, 2002.

[5] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

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

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

[8] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a fixed budget. In *Advances in NIPS 18*, pages 259–266. MIT Press, 2006.

[9] O. Dekel and O. Shamir. Learning to classify with missing and corrupted features. In *Proc. of ICML-08*, pages 216–223, 2008.

[10] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: A desktop environment to support multi-tasking knowledge workers. In *Proc. of IUI-05*, pages 75–82, 2005.

[11] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 272–279, New York, NY, USA, 2008. ACM.

[12] Y. Freund and R. E. Schapire. Game theory, on-line prediction and boosting. In *Proc. of COLT-96*, pages 325–332, 1996.

[13] C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2002.

[14] A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *Proc. of ICML-06*, pages 353–360, 2006.

[15] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[16] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

[17] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proc. of CHI-99*, pages 159–166, 1999.

[18] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *UAI-98*, pages 256–265, 1998.

[19] T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer Academic Publishers, 2001.

[20] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proc. of ICML-08*, pages 440–447, 2008.

[21] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, August 2004.

[22] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

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

[24] M. W. Mahoney, L.-H. Lim, and G. E. Carlsson. Algorithmic and statistical challenges in modern large-scale data analysis are the focus of MMDS 2008. *SIGKDD Explorations*, 10(2):57–60, 2008.

[25] A. Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proc. of ICML-04*, pages 78–85, 2004.

[26] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, and D. Hahnel. The probabilistic activity toolkit: Towards enabling activity-aware computer interfaces. Technical Report IRS-TR-03-013, Intel Research Lab, Seattle, WA, 2003.

[27] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[28] F. Rosenblatt. The Perceptron: a probabilistic model for information storage and organization in the brain. *Neurocomputing: foundations of research*, pages 89–114, 1988.

[29] J. Shen, J. Irvine, X. Bao, M. Goodman, S. Kolibaba, A. Tran, F. Carl, B. Kirschner, S. Stumpf, and T. Dietterich. Detecting and correcting user activity switches: Algorithms and interfaces. In *Proc. of IUI-09*, pages 117–125, 2009.

[30] J. Shen, L. Li, and T. G. Dietterich. Real-time detection of task switches of desktop users. In *Proc. of IJCAI-07*, pages 2868–2873, 2007.

[31] J. Shen, L. Li, T. G. Dietterich, and J. Herlocker. A hybrid learning system for recognizing user tasks from desktop activities and email messages. In *Proc. of IUI-06*, pages 86–92, 2006.

[32] S. Solla and O. Winther. Optimal perceptron learning: as online bayesian approach. In *On-line learning in neural networks*, pages 379–398. Cambridge University Press, 1998.

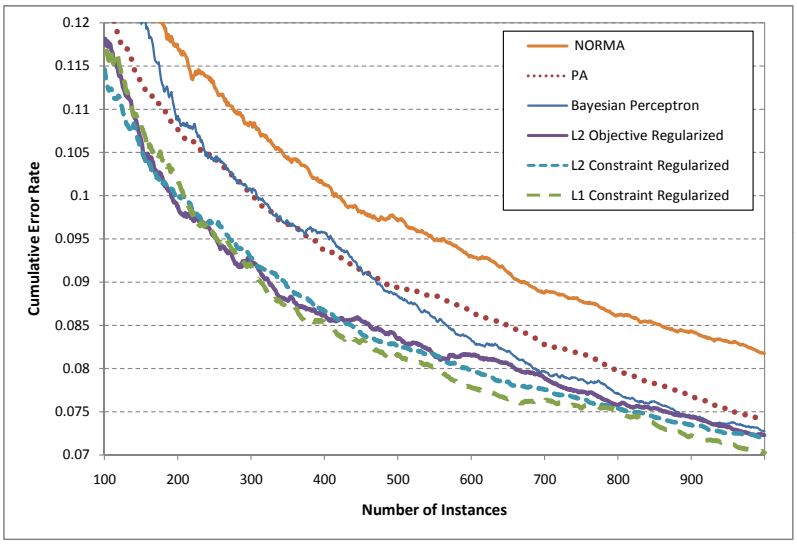
[33] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B*, 58(1):267–288, 1996.

[34] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent out-

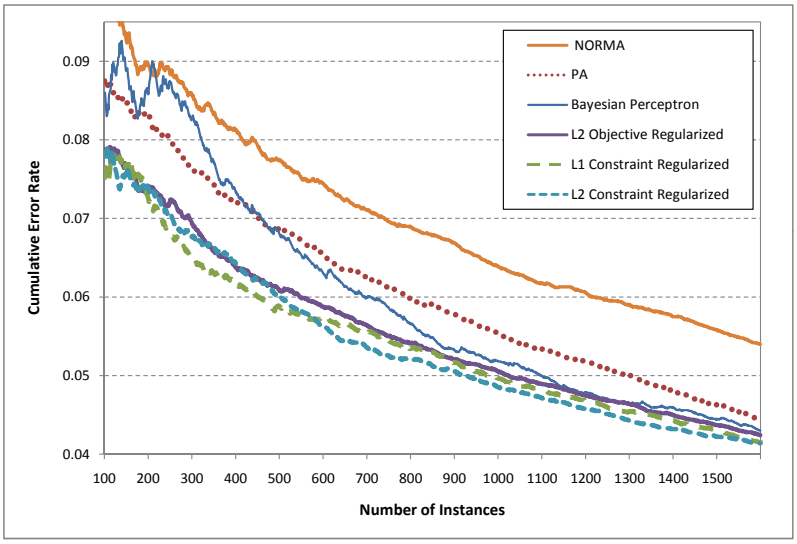
put variables. *JMLR*, 6:1453–1484, 2005.

[35] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. of ICML-97*, pages 412–420, 1997.

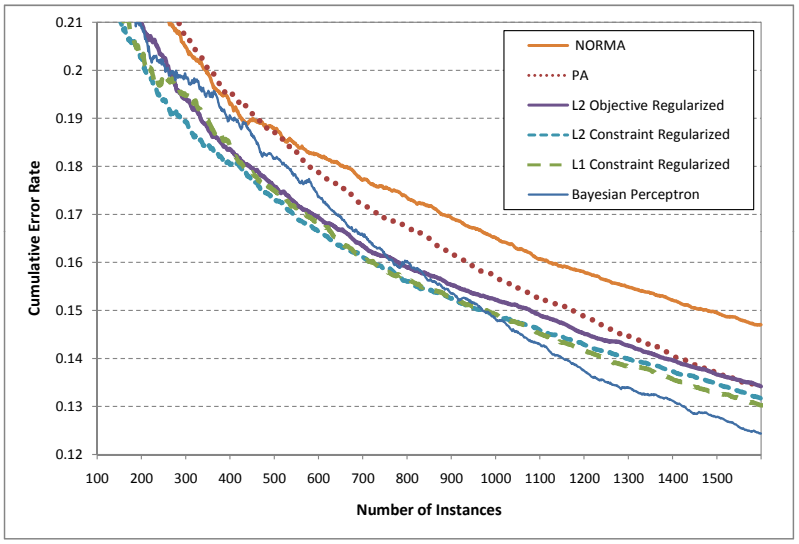
[36] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. Royal. Stat. Soc. B.*, 67(2):301–320, 2005.



(a) USPS

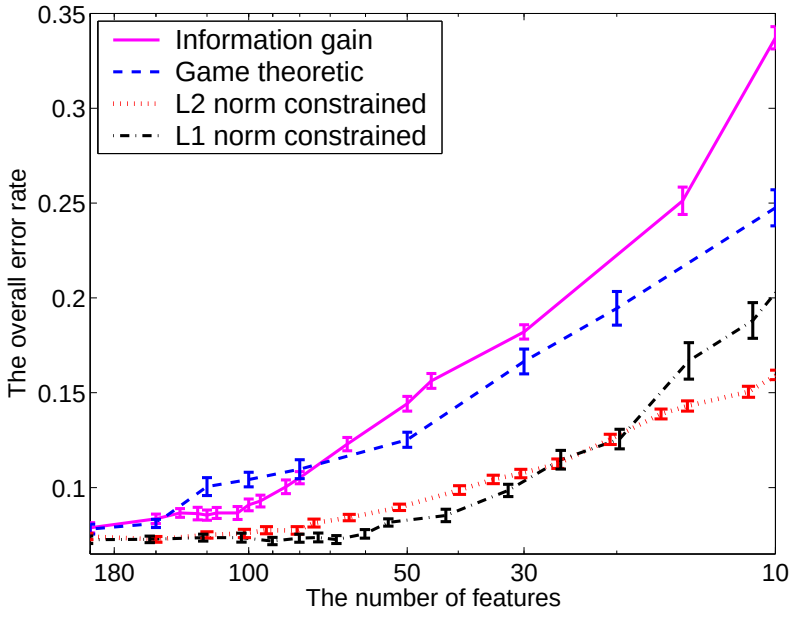


(b) 20NewsGroups

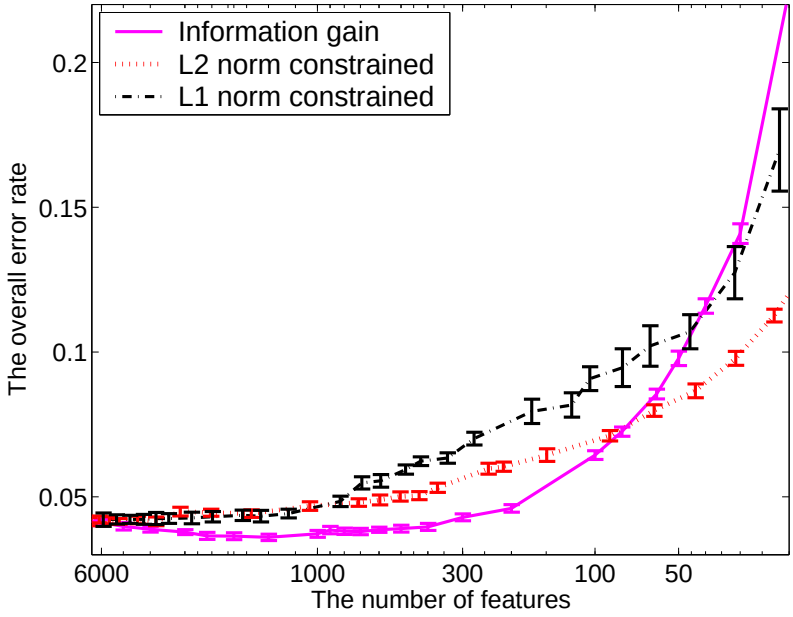


(c) SRAA

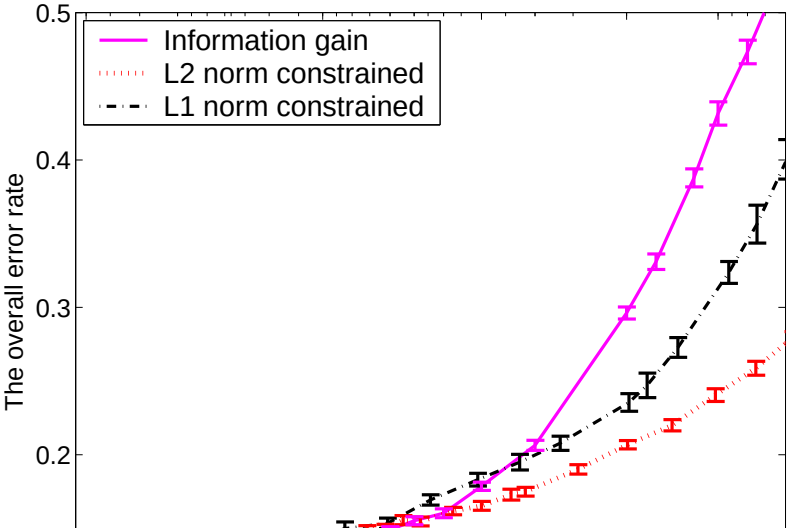
Figure 1: Cumulative error rate of different online methods as a function of the number of instances.

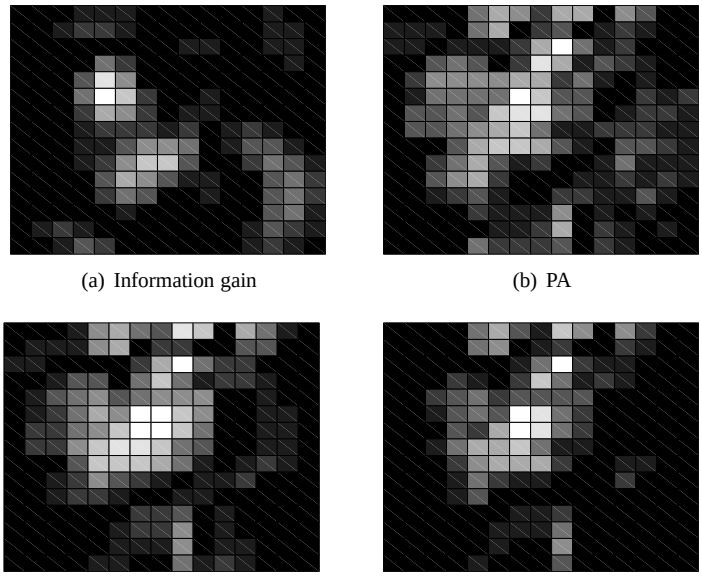


(a) USPS



(b) 20NewsGroups





(a) Information gain (b) PA

(c) L2 norm constraint (d) L1 norm constraint

Figure 4: The learned models. The lighter color indicates larger absolute value of the weights.