Online Classification Using a Voted RDA Method

Tianbing Xu
Computer Science
University of California, Irvine

Jianfeng Gao, Lin Xiao
Microsoft Research
Redmond, WA

Amelia C. Regan
Computer Science
University of California, Irvine

Abstract

We propose a voted dual averaging method for online classification problems with explicit regularization. This method employs the update rule of the regularized dual averaging (RDA) method proposed by Xiao, but only on the subsequence of training examples where a classification error is made. We derive a bound on the number of mistakes made by this method on the training set, as well as its generalization error rate. We also introduce the concept of relative strength of regularization, and show how it affects the mistake bound and generalization performance. We examine the method using ℓ_1-regularization on a large-scale natural language processing task, and obtained state-of-the-art classification performance with fairly sparse models.

Introduction

Driven by Internet applications, more and more large scale machine learning problems are emerging and require efficient online solutions. An example is online email spam filtering. Each time an email arrives, we need to decide whether it is a spam or not; after a decision is made, we may receive the true value feedback information from users, and thus update the hypothesis and continue the classification in an online fashion. The low computational cost of online methods such as stochastic gradient descent is associated with their slow convergence rate, which effectively introduces implicit regularization and is possible to prevent overfitting for very large scale training data (Zhang 2004; Bottou and Bousquet 2008).

To obtain better generalization performance, or to induce a particular structure (such as sparsity) into the solution, it is often desirable to add simple regularization terms to the loss function of a learning problem. In the online setting, Langford et al. (Langford, Li, and Zhang 2009) proposed a *truncated gradient* method to induce sparsity in the online gradient method for minimizing convex loss functions with ℓ_1-regularization, Duchi and Singer (Duchi and Singer 2009) applied forward-backward splitting method to work with more general regularizations, and Xiao (Xiao 2010) extended Nesterov's work (Nesterov 2009) to develop regularized dual averaging (RDA) methods. In the case of ℓ_1 regularization, RDA often generates significantly more sparse solutions than other online methods, which match the sparsity results of batch optimization methods. Recently, Lee and Wright (Lee and Wrigh 2012) show that under suitable conditions, RDA is able to identify the low-dimensional sparse manifold with high probability.

The aforementioned work provide regret analysis or convergence rate in terms of reducing the objective function in a convex optimization framework. For classification problems, such an objective function is a weighted sum of a loss function (such as hinge or logistic loss) and a regularization term (such as ℓ_2 or ℓ_1 norm). Since the loss function is a convex surrogate for the 0-1 loss, it is often possible to derive a classification error bound based on their regret bound or convergence rate. However, this connection between regret bound and error rate can be obfuscated by the additional regularization term.

In this paper, we propose a *voted RDA* (vRDA) method for regularized online classification, and derive its error bounds (i.e., number of mistakes made on the training set), as well as its generalization performance. We also introduce the concept of *relative strength of regularization*, and show how it affects the error bound and generalization performance.

The voted RDA method shares a similar structure as the *voted perceptron* algorithm (Freund and Schapire 1999), which is a combination of the *perceptron* algorithm (Rosenblatt 1958) and the leave-one-out online-to-batch conversion method (Helmbold and Warmuth 1995). More specifically, in the training phase, we perform the update of the RDA method only on the subsequence of examples where a prediction mistake is made. In the testing phase, we follow the deterministic leave-one-out approach, which labels an unseen example with the majority voting of all the predictors generated in the training phase. In particular, each predictor is weighted by the number of examples it survived to predict correctly in the training phase.

The key difference between the voted RDA method and the original RDA method (Xiao 2010) is that voted RDA only updates its predictor when there is a classification error. In addition to numerous advantages in terms of computational learning theory (Floyd and Warmuth 1995), it can reduce the computational cost involved in updating the predictor. Moreover, the scheme of update only on errors allows us to derive a bound on the number of classification errors that does not depend on the total number of examples.

Our analysis on the number of mistakes made by the al-

gorithm is based on the regret analysis of the RDA method (Xiao 2010). The result depends on the *relative strength of regularization*, which is captured by the difference between the size of the regularization term of an (unknown) optimal predictor, and the average size of the online predictors generated by the voted RDA method. In absense of the regularization term, our results matches that of the voted perceptron algorithm (up to a small constant). Moreover, our notion of relative strength of regularization and error bound analysis also applies to the voted versions of other online algorithms, including the forward-backward splitting method (Duchi and Singer 2009).

Regularized Online Classification

In this paper, we mainly consider binary classification problems. Let $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a set of training examples, where each example consists of a feature vector $x_i \in \mathbb{R}^n$ and a label $y_i \in \{+1, -1\}$. Our goal is to learn a classification function $f : \mathbb{R}^n \to \{+1, -1\}$ that attains a small number of classification errors. For simplicity, we focus on the linear predictor

$$f(w, x) = \mathrm{sign}(w^T x),$$

where $w \in \mathbb{R}^n$ is a weight vector, or *predictor*.

In a batch learning setting, we find the optimal predictor w that minimizes the following empirical risk

$$R_{\mathrm{emp}}(w) = \frac{1}{m} \sum_{i=1}^{m} \ell(w, z_i) + \lambda \Psi(w),$$

where $\ell(w, z_i)$ is a loss function at sample $z_i = (x_i, y_i)$, and $\Psi(w)$ is a regularization function to prevent overfitting or induce a particular structure (e.g., ℓ_1 norm for sparsity). If we use the 0-1 loss function

$$\ell(w, z) = 1\Big(y = f(w, x)\Big) = \begin{cases} 1 & \text{if } y = f(w, x) \\ 0 & \text{otherwise} \end{cases}$$

then the total loss $\sum_{i=1}^{m} \ell(w, z_i)$ is precisely the total number of classification errors made by the predictor w.

However, the 0-1 loss function is non-convex and thus it is very difficult to optimize. In practice, we often use a surrogate convex function, such as the *hinge loss* $\ell(w, z) = \max\{0, (1 - yw^T x)\}$, the *logistic loss* $\ell(w, z) = \log_2(1 + \exp(-yw^T x))$, or the *exponential loss* $\ell(w, z) = \exp(-yw^T x)$. We note that these surrogate functions are upper bounds of the 0-1 loss, therefore the corresponding total loss $\sum_{i=1}^{m} \ell(w, z)$ is an upper bound on the total number of classification errors.

In a online classification setting, the training examples $\{z_1, z_2, \ldots, z_t, \ldots\}$ are given one at a time, and accordingly, we generate a sequence of hypotheses w_t one at a time. At each time t, we make a prediction $f(w_t, x_t)$ based on the previous hypothesis w_t, then calculate the loss $\ell(w_t, z_t)$ based on the true label y_t. The next hypothesis w_{t+1} is updated according to some rules, e.g., online gradient descent (Zinkevich 2003), based on the information available up to time t. To simplify notation in the online setting, we use a subscript to indicate the loss function at time t, i.e., we write $\ell_t(w_t) = \ell(w_t, z_t)$ henceforth.

The Voted RDA Method

Algorithm 1 The voted RDA method (training)

input: training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, and number of epochs N
initialize: $k \leftarrow 1,\ w_1 \leftarrow 0,\ c_1 \leftarrow 0,\ s_0 \leftarrow 0$
repeat
  for $i = 1, \ldots, m$ **do**
    compute prediction: $\hat{y} \leftarrow f(w_k, x_i)$
    if $\hat{y} = y_i$ **then**
      $c_k \leftarrow c_k + 1$
    else
      compute subgradient $g_k \in \partial \ell_i(w_k)$
      $s_k \leftarrow s_{k-1} + g_k$
      update w_{k+1} according to Eq. (1)
      $c_{k+1} \leftarrow 1$
      $k \leftarrow k + 1$
    end if
  end for
until N times
output: number of mistakes M, and a list of predictors $\{(w_1, c_1), \ldots, (w_M, c_M)\}$

The voted RDA method is described in Algorithm 1 and Algorithm 2, for training and testing respectively. The structure of the algorithm description is very similar to the voted perceptron algorithm (Freund and Schapire 1999). In the training phase (Algorithm 1), we go through the training set N times, and only update the predictor when it makes a classification error. Each predictor w_k is associated with a counter c_k, which counts the number of examples it processed correctly. These counters are then used in the testing module (Algorithm 2) as the voting weights to generate a prediction on an unlabeled example.

The update rule used in Algorithm 1 takes the same form as the RDA method (Xiao 2010):

$$w_{k+1} = \arg\min_{w} \left\{ \frac{1}{k} s_k^T w + \lambda \Psi(w) + \frac{\beta_k}{k} h(w) \right\}, \quad (1)$$

where $\Psi(w)$ is the convex regularization function, $h(w)$ is an auxiliary strongly convex function, and

$$\beta_k = \eta \sqrt{k}, \qquad \forall\, k \geq 1, \quad (2)$$

where $\eta > 0$ is a parameter that controls the learning rate. Note that k is the number of classification mistakes, s_k is the summation of subgradients for the k samples with classification mistakes, and c_k is the counter of survival times for the predictor w_k.

For large scale problems, storing the list of predictors $\{(w_1, c_1), \ldots, (w_M, c_M)\}$ and computing the majority vote in (3) can be very costly. For linear predictors (i.e., $\hat{y} = \mathrm{sign}(w^T x)$), we can replace the majority vote with a single prediction made by the weighted average predictor $\tilde{w}_M = (1/M) \sum_{k=1}^{M} c_k w_k$,

$$\hat{y} = \mathrm{sign}\left(\tilde{w}_M^T x\right) = \mathrm{sign}\left(\frac{1}{M} \sum_{k=1}^{M} c_k (w_k^T x)\right).$$

Algorithm 2 The voted RDA method (testing)

given: weighted predictors $\{(w_1, c_1), \ldots, (w_M, c_M)\}$
input: an unlabeled instance x
output: a predicted label $\hat{y}$ given by:

$$\hat{y} = \text{sign}\left(\sum_{k=1}^{M} c_k\, f(w_k, x)\right) \tag{3}$$

In practice, this weighted average predictor generates very similar robust performance as the majority vote (Freund and Schapire 1999), and saves lots of memory and computational cost.

Bound On The Number Of Mistakes

We provide an analysis of the voted RDA method for the case $N = 1$ (i.e., going through the training set once). The analysis parallels that for the voted perceptron algorithm given in Freund and Schapire (Freund and Schapire 1999). In this section, we bound the number of mistakes made by the voted RDA method through its regret analysis. Then in the next section, we give its expected error rate in an online-to-batch conversion setting.

First, we recognize that the voted RDA method is equivalent to running the RDA method (Xiao 2010) on the subsequence of training examples where a classification mistake is made. Let M the number of prediction mistakes made by the algorithm after processing the m training examples, and $i(k)$ denote the index of the example on which the k-th mistake was made (by w_k). The regret of the algorithm, with respect to a fixed vector w, is defined only on the examples with prediction errors:

$$R_M(w) = \sum_{k=1}^{M}\big(\ell_{i(k)}(w_k)+\Psi(w_k)\big)-\sum_{k=1}^{M}\big(\ell_{i(k)}(w)+\Psi(w)\big). \tag{4}$$

According to Theorem 1 of Xiao (Xiao 2010), the RDA method (1) has the following regret bound:

$$R_M(w) \le \beta_M h(w) + \frac{G^2}{2}\sum_{k=1}^{M}\frac{1}{\beta_k},$$

where G is an upper bound on the norm of the subgradients, i.e., $\|g_k\|_2 \le G$ for all $k = 1, \ldots, M$. For simplicity of presentation, we restrict to the case of $h(w) = (1/2)\|w\|_2^2$ in this paper. If we choose β_k as in (2), then, by Corollary 2 of Xiao (Xiao 2010),

$$R_M(w) \le \left(\frac{\eta}{2}\|w\|_2^2 + \frac{G^2}{\eta}\right)\sqrt{M}.$$

This bound is minimized by setting $\eta = \sqrt{2}G/\|w\|_2$, which results in

$$R_M(w) \le \sqrt{2}G\|w\|_2\sqrt{M}. \tag{5}$$

To bound the number of mistakes M, we use the fact that the loss functions $\ell_i(w)$ are surrogate (upper bounds) for the 0-1 loss. Therefore,

$$M \le \sum_{k=1}^{M}\ell_{i(k)}(w_k).$$

Combining the above inequality with the definition of regret in (4) and the regret bound (5), we have

$$M \le \sum_{k=1}^{M}\ell_{i(k)}(w) + M\lambda\Delta(w) + \sqrt{2}G\|w\|_2\sqrt{M}. \tag{6}$$

where $\Delta(w)$ is the *relative strength of regularization*, defined as

$$\lambda\Delta(w) = \Psi(w) - \frac{1}{M}\sum_{k=1}^{M}\Psi(w_k). \tag{7}$$

We can also further relax the bound by replacing $\Delta(w)$ with $\bar{\Delta}(w)$, defined as

$$\lambda\bar{\Delta}(w) = \Psi(w) - \Psi(\bar{w}_M),$$

where $\bar{w}_M = \frac{1}{M}\sum_{k=1}^{M} w_k$ is the (unweighted) average of the predictors generated by the algorithm. Note that by convexity of Ψ, we have $\Delta(w) \le \bar{\Delta}(w)$.

Analysis For The Separable Case

Our analysis for the separable case is based on the hinge loss $\ell_i(w) = \max\{0,\, 1 - y_i(w^T x_i)\}$.

Assumption 1 *There exists a vector u such that $y_i(u^T x_i) \ge 1$ for all $i = 1, \ldots, m$.*

This is the standard *separability with large margin* assumption. Under this assumption, we have

$$\sum_{k=1}^{M}\ell_{i(k)}(u) = \sum_{k=1}^{M}\max\{0, 1 - y_{i(k)}(u^T x_{i(k)})\} = 0$$

for any $M > 0$ and any subsequence $\{i(k)\}_{i=1}^{M}$. The margin of separability is defined as $\gamma = 1/\|u\|_2$. For convenience, we also let

$$R = \max_{i=1,\ldots,m}\|x_i\|_2\,.$$

Then we can set $G = R$ since for hinge loss, $-y_i x_i$ is the subgradient of $\ell_i(w)$, and we have $\|-y_i x_i\|_2 = \|x_i\|_2 \le R$ for $i = 1, \ldots, m$. We have the following results under Assumption 1:

If $\lambda = 0$ (the case without regularization), then $M \le \sqrt{2}G\|u\|_2\sqrt{M}$, which implies

$$M \le 2G^2\|u\|_2^2 = 2\left(\frac{R}{\gamma}\right)^2.$$

This is very similar to the mistake bound for the voted perceptron (Freund and Schapire 1999), with an extra factor of two. Note that this bound is independent of the dimension n and the number of examples m. It also holds for $N > 1$ (multiple passes over the data).

If $\lambda > 0$, the mistake bound also depends on $\Delta(u)$, which is the difference between $\Psi(u)$ and the *unweighted* average of $\Psi(w_1), \ldots, \Psi(w_M)$. More specifically,

$$M \le M\lambda\Delta(u) + \sqrt{2}R\|u\|_2\sqrt{M}. \tag{8}$$

Note that $\Psi(w_1), \ldots, \Psi(w_M)$ tend to be small for large values of λ (more regularization), and tend to be large for small values of λ (less regularization). We discuss two scenarios:

The under-regularization case: $\Delta(u) < 0$. This happens if the regularization parameter λ is chosen too small, and the generated vectors $w_1, \ldots, w_M$ on average has a larger value of Ψ than $\Psi(u)$. In this case, we have

$$M \leq 2 \left(\frac{1}{1 + \lambda |\Delta(u)|} \right)^2 \left(\frac{R}{\gamma} \right)^2.$$

So we have a smaller mistake bound than the case of "perfect" regularization (when $\Delta(u) = 0$). This effect may be related to over-fitting on the training set.

The over-regularization case: $\Delta(u) > 0$. This happens if the regularization parameter λ is chosen too large, and the generated vectors $w_1, \ldots, w_M$ on average has a smaller Ψ value than $\Psi(u)$. If in addition $\lambda |\Delta(u)| < 1$, then we have

$$M \leq 2 \left(\frac{1}{1 - \lambda |\Delta(u)|} \right)^2 \left(\frac{R}{\gamma} \right)^2,$$

which can be much larger than the case of "perfect" regularization (meaning $\Delta(u) = 0$). If $\lambda \Delta(u) \geq 1$, then the inequality (8) holds trivially and does not give any meaningful mistake bound.

Analysis For The Inseparable Case

We start with the inequality (6). To simplify notation, let $L(u)$ denote the total loss of an arbitrary vector u over the subsequence $i(k)$, $k = 1, \ldots, M$, i.e.,

$$L(u) = \sum_{k=1}^{M} \ell_{i(k)}(u). \tag{9}$$

Then we have

$$M \leq L(u) + M \lambda \Delta(u) + \sqrt{2} R \|u\|_2 \sqrt{M}. \tag{10}$$

Our analysis is similar to the error analysis for the perceptron in (Shalev-Shwartz 2011).

If $\lambda = 0$ (the case without regularization), we have

$$M \ \leq \ L(u) + \sqrt{2} R \|u\|_2 \sqrt{M},$$

which results in

$$M \ \leq \ \left(\sqrt{L(u)} + \sqrt{2} R \|u\|_2 \right)^2.$$

Note that this bound only makes sense if the total loss $L(u)$ is not too large.

If $\lambda > 0$, the mistake bound depends on $\Delta(u)$, the relative strength of regularization.

The under-regularization case: $\Delta(u) < 0$. we have

$$M \ \leq \ \left(\sqrt{\frac{L(u)}{1 + \lambda |\Delta(u)|}} + \frac{\sqrt{2} R \|u\|_2}{1 + \lambda |\Delta(u)|} \right)^2.$$

The over-regularization case: $\Delta(u) > 0$. If $\lambda |\Delta(u)| < 1$, we have

$$M \ \leq \ \left(\sqrt{\frac{L(u)}{1 - \lambda |\Delta(u)|}} + \frac{\sqrt{2} R \|u\|_2}{1 - \lambda |\Delta(u)|} \right)^2.$$

Again, if $\lambda \Delta(u) \geq 1$, the inequality (10) holds trivially and does not lead to any meaningful bound.

Theorem 1 *Let $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a sequence of labeled examples with $\|x_i\|_2 \leq R$. Suppose the voted RDA method (Algorithm 1) makes M prediction errors on the subsequence $i(1), \ldots, i(M)$, and generates a sequence of predictors $w_1, \ldots, w_M$. For any vector u, let $L(u)$ be the total loss defined in (9), and $\Delta(u)$ be the relative strength of regularization defined in (7). If $\lambda \Delta(u) < 1$, then the number of mistakes M is bounded by*

$$M \ \leq \ \left(\sqrt{\frac{L(u)}{1 - \lambda \Delta(u)}} + \frac{\sqrt{2} R \|u\|_2}{1 - \lambda \Delta(u)} \right)^2.$$

In particular, if the training set satisfies Assumption 1, then we have

$$M \ \leq \ 2 \left(\frac{1}{1 - \lambda \Delta(u)} \right)^2 \left(\frac{R}{\gamma} \right)^2,$$

where $\gamma = 1/\|u\|_2$ is the separation margin.

The above theorem is stated in the context of using the hinge loss. However, the analysis for the inseparable case holds for other convex surrogate functions as well, including the hinge loss, logistic loss and exponential loss. We only need to replace R with a constant G, which satisfies $G \geq \|g_k\|_2$ for all $k = 1, \ldots, M$.

For a strongly convex regularizer such as $\Psi(w) = (\lambda/2)\|w\|_2^2$, the regret bound is on the order of $\log M$ (Xiao 2010). Thus, for any hypothesis u, the training error bound can be derived from

$$M(1 - \lambda \Delta(u)) \leq G \|u\|_2 \log M + L(u).$$

Online SVM is a special case following the above bound with hinge loss and ℓ_2 regularizer.

Online-To-Batch Conversion

The training part of the voted RDA method (Algorithm 1) is an online algorithm, which makes a small number of mistakes when presented with examples one by one (see the analysis in Section). In a batch setting, we can use this algorithm to process the training data one by one (possibly going through the data multiple times), and then generate a hypothesis which will be evaluated on a separate test set.

Following Freund and Schapire (Freund and Schapire 1999), we use the deterministic leave-one-out method for converting an online learning algorithm into a batch learning algorithm. Here we give a brief description. Suppose we have m training examples and an unlabeled instance, all generated i.i.d. at random. Then, for each $r \in \{0, m\}$, we run the online algorithm on a sequence of $r + 1$ examples consisting of the first r examples in the training set and the last one being the unlabeled instance. This produces $m + 1$ predictions for the unlabeled instance, and we take the majority vote of these predictions.

It is straightforward to see that the testing module of the voted RDA method (Algorithm 2) outputs exactly such a majority vote, hence the name "voted RDA." Our result is a direct corollary of a theorem from Freund and Schapire (Freund and Schapire 1999), which is a result of the theory developed in Helmbold and Warmuth (Helmbold and Warmuth 1995).

Table 1: Comparing performance of different algorithms

Algorithms	Precision	Recall	F-Score	NNZ
Baseline	0.8983	0.8990	0.8986	N.A.
Perceptron	0.9191	0.9143	**0.9164**	**939 K**
TG (hinge)	0.9198	0.9127	**0.9172**	**775 K**
TG (log)	0.9190	0.9139	**0.9165**	**485 K**
vRDA (hinge)	0.9211	0.9150	**0.9175**	**932 K**
vRDA (log)	0.9204	0.9144	**0.9174**	**173 K**

Corollary 1 *Assume all examples are generated i.i.d. at random. Suppose that we run Algorithm 1 on a sequence of examples* $\{(x_1, y_1), \ldots, (x_{m+1}, y_{m+1})\}$ *and* M *mistakes occur on examples with indices* $i(1), \ldots, i(M)$*. Let* $\Delta(u)$ *and* $L(u)$ *be defined as in (7) and (9), respectively.*

Now suppose we run Algorithm 1 on m *examples* $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ *for a single epoch. Then the probability that Algorithm 2 does not predict* y_{m+1} *on the test instance* x_{m+1} *is at most*

$$\frac{2}{m+1}\,\mathbb{E}\left[\inf_{u:\,1-\lambda\Delta(u)>0}\left(\sqrt{\frac{L(u)}{1-\lambda\Delta(u)}}+\frac{\sqrt{2}R\|u\|_2}{1-\lambda\Delta(u)}\right)^2\right].$$

(The above expectation $\mathbb{E}[\cdot]$ *is over the choice of all* $m+1$ *random examples.)*

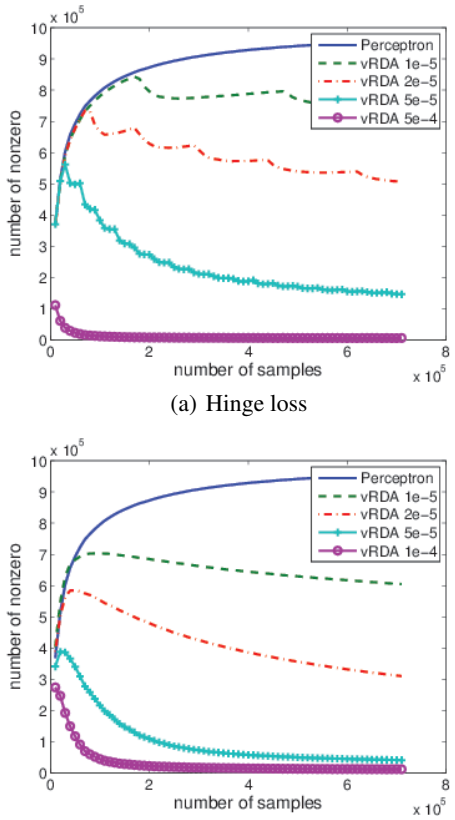
Experiments On Parse Reranking

Parse reranking has been widely used as a test bed when adapting machine learning algorithms to natural language processing (NLP) tasks; see, e.g., Collins (Collins 2000), Charniak and Johnson (Charniak and Johnson 2005), Gao et al. (Gao et al. 2007) and Andrew and Gao (Andrew and Gao 2007). Here, we briefly describe parse reranking as an online classification problem, following Collins (Collins 2000).

We follow the experimental paradigm of parse reranking outlined in Charniak and Johnson (Charniak and Johnson 2005). We used the same generative baseline model for generating candidate parsers, and nearly the same feature set, which includes the log probability of a parse according to the baseline model and 1,219,272 additional features. We trained the predictor on Sections 2-19 of the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993), used Section 20-21 to optimize training parameters, including the regularization weight λ and the learning rate η, and then evaluated the predictors on Section 22. The training set contains 36K sentences, while the development set and the test set have 4K and 1.7K, respectively. Performance of parsing reranking is measured with the PARSEVAL metric, i.e., F-Score over labelled brackets. For each epoch, we have the F-Score based on the corresponding weights learned from these samples. We use the weighted average of all the predictors generated by the algorithm as the final predictor for testing.

Comparison with Perceptron and TG

Our main results are summarized in Tables 1, the F-Score and NNZ are averaged over the results of 20 epoches of online classification. The baseline results are obtained by



(a) Hinge loss

(b) Log loss

Figure 1: Different sparse feature structure by different regularization λ for vRDA with hinge and log losses. The x axis is the number of samples, and the y axis shows the NNZ.

the parser in Charniak (Charniak 2000). The implementation of perceptron follows the averaged perceptron algorithm (Collins 2002). For voted RDA, we report results of the predictors trained using the parameter settings tuned on the development set. We used $\eta = 0.05$ and $\lambda = 1e-5$ for hinge loss, and $\eta = 1000$ and $\lambda = 1e-4$ for log loss. Results show that compared to perceptron, voted RDA achieves similar F-Scores with more sparse weight vectors. For example, using log loss we are able to achieve an F-score of 0.9174 with only 14% of features. TG is the truncated gradient method (Langford, Li, and Zhang 2009); our vRDA is a better choice than TG in terms of the classification performance and sparsity, especially for log loss.

Sparsity and Performance Trade Off

Since its ability to learn a sparse structured weight vector is an important advantage of voted RDA, we exam in detail how the number of non-zero weights changes during the course of training in Figure 1. In vRDA, the regularization parameter λ controls the model sparsity. For a stronger ℓ_1 regularizer with large values of λ, it ends up with a simpler model with fewer number of nonzero (NNZ) feature weights; for a weaker regularizer, we will get a more complex model with many more nonzero features weights. From the Figure 1, we may observe the convergence of the online

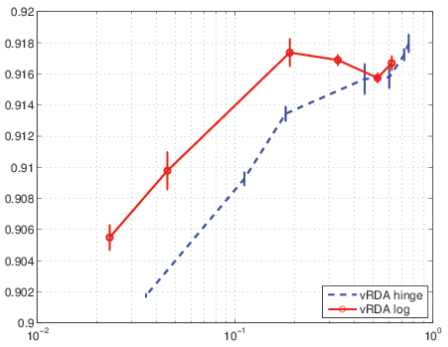


Figure 2: Trade off between the model sparsity and classification accuracy for vRDA with hinge and log losses. The x axis is the ratio of number of nonzero to the overall 1.2 M features; y axis is the F-Score.

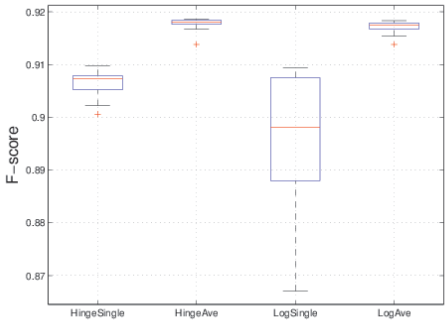


Figure 3: Performance comparisons of single and average predctions for vRDA.

learning along with the number of samples. With a relatively larger value of λ, the simpler model is easy to converge to stationary states with a small number of nonzero feature weights; while for a smaller λ, we have more nonzero feature weights and it will take many more samples for the model to reach stable states.

Figure 2 illustrates the trade-off between model sparsity and classification performance when we adjust the regularization parameter λ. For hinge loss, with a larger λ, we get more sparse model at the price of a worse F-Score. For the log loss, as is showed in Figure 2, it is able to prevent overfitting to some extent. On average, it achieves the best classification performance with average F-Score 0.9174 with the 173K (out of 1.2M) feature chosen by the sparse predictor.

Single vs Average Prediction

To investigate where the performance gain comes from, we compare the predictions of vRDA using a single weight at the last sample of each epoch, and the averaged weights learned from all the training samples. In Figure 3, we plot the mean and variance bars with the corresponding predictions based on weights trained on 10 epoches. For both Hinge and Log losses, the average predictions have lower variance and better F-Score compared to their single predictions. The large variance for single predictions of Log loss implies that the predictions are quite inconsistent by different epoches; thus average predictions is highly desired here.

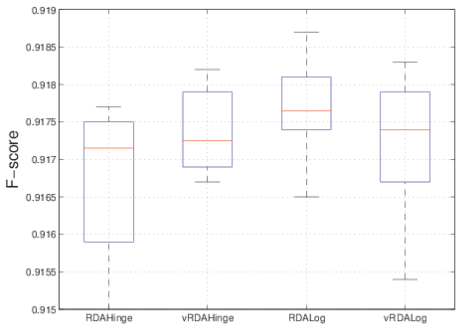


Figure 4: Performance comparisons of RDA and vRDA.

Conservative Updates

Here, in Figure 4 we compare the performance of RDA and vRDA to illustrate the trade-off with conservative updates with mean and variance bars based on 10 epoches. For Hinge loss, the conservative updates (vRDA) are necessary as the hinge loss is nonzero when there is a classification mistake; thus vRDA has better F-Scores. While for Log loss, RDA is better as even when there is a classification mistake, we still have a non-zero loss and need to update the weights accordingly. Another gain by conservative updates comes from a computational perspective. For vRDA, the frequency ratio of updating weights is proportional to the error rate of RDA. From our experiments, the training time of vRDA is about 89.7% for Hinge loss and 87.2% for Log loss of RDA. These percentages are not the error rate as there are extra common computations involved in calculating that.

Conclusion And Discussions

In this paper, we propose a voted RDA (vRDA) method to address online classification problems with explicit regularization. This method updates the predictor only on the subsequence of training examples where a classification error is made. In addition to significantly reducing the computational cost involved in updating the predictor, this allows us to derive a mistake bound that does not depend on the total number of examples. We also introduce the concept of relative strength of regularization, and show how it affects the mistake bound and the generalization performance. Finally, our algorithm obtained state-of-the-art classification performance with fairly sparse models for parse reranking task.

Our analysis on mistake bound is based on the regret analysis of the RDA method (Xiao 2010). In fact, our notion of relative strength of regularization and error bound analysis also applies to the voted versions of other online algorithms that admit a similar regret analysis, including the forward-backward splitting method in (Duchi and Singer 2009).

Acknowledgments

This work was supported by Microsoft Research internship and a UCI ICS Deans Fellowship. Special thanks for insightful discussions with Guibo Ye and Qiang Liu.

References

Andrew, G., and Gao, J. 2007. Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine learning(ICML 07)*, 33–40.

Bottou, L., and Bousquet, O. 2008. The tradeoffs of large scale learning. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press. 161–168.

Charniak, E., and Johnson, M. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics(ACL 05)*, 173–180.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL 2000)*, 132–139.

Collins, M. 2000. Discriminative re-ranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine learning(ICML)*, 175–182.

Collins, M. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing*, 1–8.

Duchi, J., and Singer, Y. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research* 10:2873–2898.

Floyd, S., and Warmuth, M. K. 1995. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning* 21(3):269–304.

Freund, Y., and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. *Machine Learning* 37(3):227–296.

Gao, J.; Andrew, G.; Johnson, M.; and Toutanova, K. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 824–831.

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

Langford, J.; Li, L.; and Zhang, T. 2009. Sparse online learning via truncated gradient. *Journal of Machine Learning Research* 10:777–801.

Lee, S., and Wrigh, S. J. 2012. Manifold identication in dual averaging for regularized stochastic online learning. *Journal of Machine Learning Research* 13:1705–1744.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Nesterov, Y. 2009. Primal-dual subgradient methods for convex problems. *Mathematical Programming* 120:221–259.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65:386–407.

Shalev-Shwartz, S. 2011. Online learning and online convex optimization. *Foundations and Trends in Machine Learning* 4(2):107–194.

Xiao, L. 2010. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research* 11:2543–2596.

Zhang, T. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine learning (ICML 04)*, 116–123.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML03)*, 928–936.