A Voted Regularized Dual Averaging Method for Large-Scale Discriminative Training in Natural Language Processing

Jianfeng Gao
Microsoft Research, Redmond
Washington 98052, USA
jfgao@microsoft.com

Tianbing Xu
University of California, Irvine
California 92697, USA
tbing.xu@gmail.com

Lin Xiao
Microsoft Research, Redmond
Washington 98052, USA
Lin.Xiao@microsoft.com

Xiaodong He
Microsoft Research, Redmond
Washington 98052, USA
xiaohe@microsoft.com

Abstract

We propose a new algorithm based on the dual averaging method for large-scale discriminative training in natural language processing (NLP), as an alternative to the perceptron algorithms or stochastic gradient descent (SGD). The new algorithm estimates parameters of linear models by minimizing L_1 regularized objectives and are effective in obtaining sparse solutions, which is particularly desirable for large scale NLP tasks. We then give the mistake bound of the algorithm, and show how the bound is affected by the additional L_1 regularization term. Evaluations on the tasks of parse reranking and statistical machine translation attest the success of the new algorithm.

1 Introduction

The perceptron algorithm and its variants have proved to be effective for discriminative training in many natural language processing (NLP) problems, such as language modeling (Roark et al. 2007), parsing (Collins 2002b), and statistical machine translation (SMT) (Shen et al. 2004; Liang et al. 2006). The popularity of perceptron is mainly due to its competitive performance, simplicity in implementation and low computational cost in training, compared to the batch training methods such as maximum entropy estimation (Collins 2002a; Gao et al. 2007). However, recent results on applying perceptron to large-scale discriminative training for SMT motivate researchers to revisit its limitations by seeking improvements or alternative methods. For example, Gimpel and Smith (2012) proposed an alternative algorithm that can optimize a particular family of ramp loss functions tailored to SMT. To obtain better generalization performance, Simianer et al. (2012) introduced an explicit feature selection step using L_1 regularization to obtain sparse solutions. Along the same line of research, Martins et al. (2011a; 2011b) and Tsuruoka et al. (2009) proposed to use the methods similar to the truncated gradient method (Langford et al. 2009) where an L_1 regularization term is added explicitly to the loss function of a learning problem.

In this paper we propose a new parameter estimation algorithm that is a natural alternative to stochastic gradient descent (SGD) and the perceptron algorithms. The new algorithm, called voted regularized dual averaging (or VRDA), is based on the dual averaging method (Nesterov 2009), and its regularized version, the regularized dual averaging (RDA) method (Xiao 2010). The VRDA algorithm shares a similar structure as the voted perceptron algorithm (Freund and Schapire 1999). However, VRDA induces sparsity into the solutions by minimizing loss functions with L_1 regularization, and thus generates significantly sparser models than perceptron. Sparse solution is particularly desirable for large-scale NLP tasks not only for better generalization performance but also for easy deployment of the model.

The key difference between VRDA and the original dual averaging methods is that VRDA only updates its parameter vector when there is a prediction error. In addition to numerous advantages in terms of computational learning theory (Floyd and

Warmuth 1995), it can significantly reduce the computational cost involved in updating the predictor. Moreover, the scheme of update-only-on-errors allows us to derive an error bound that matches that of the voted perceptron algorithm (up to a small constant), and in addition show how the additional L_1 regularization term affects the error bound and generalization performance.

We evaluate the performance of VRDA on two large-scale NLP tasks, parse reranking and discriminative training of phrase translation models. The results show that VRDA gives better and sparser solutions compared to the perceptron algorithms or SGD and that VRDA can generate much sparser models than the truncated gradient method which is the state-of-the-art method of sparse learning for linear models.

2 Notation and Background

The two tasks studied in this paper are based on (log-)linear models (Collins 2000) which require learning a mapping between inputs $x \in X$ to outputs $y \in Y$. We are given

- Training samples (x_i, y_i) for $i = 1 \ldots m$, each x_i is labeled by a reference output y_i;
- A procedure GEN, which generates a set of N-best candidates $\mathrm{GEN}(x_i)$ for an input x_i;
- A feature mapping $\phi: X \times Y \to \mathbb{R}^d$, which maps each (x_i, y), where $y \in \mathrm{GEN}(x_i)$, to a vector of feature values; and
- A parameter vector (or *predictor*) $w \in \mathbb{R}^d$, which assigns a real-valued weight to each feature.

The components of GEN, ϕ and w define a linear model that maps x_i to an output $F(x_i)$ as follows

$$F(x_i) = \hat{y} = \underset{y \in \mathrm{GEN}(x_i)}{\operatorname{argmax}} w^T \phi(x_i, y) \qquad (1)$$

In the parse reranking task, training samples are sentence and gold-standard parse pairs. In SMT, the training samples are source sentence and reference translation pairs. In complex NLP tasks such as parsing and translation, the reference output y_i is often not guaranteed to be included in the N-best list $\mathrm{GEN}(x_i)$ even with a very large value of N. Therefore, it is a common practice to replace y_i with its oracle candidate y^* for model training (Liang et al. 2006). y^* is defined as the candidate with the lowest cost compared to its reference y_i:

$$y^* = \underset{y \in \mathrm{GEN}(x_i)}{\operatorname{argmin}} \mathrm{cost}(y_i, y), \qquad (2)$$

where $\mathrm{cost}(.)$ is an application-specific cost function. For example, $\mathrm{cost}(.)$ in SMT is typically defined as the negative sentence-level BLEU score of y compared to the reference translation y_i.

SMT involves hidden-variable models such that a hidden variable h is assumed to be constructed during the process of generating y. In phrase-based SMT, h consists of a segmentation of the source and target sentences into phrases and an alignment between source and target phrases. Thus, the linear model (1) can be rewritten, for SMT, as

$$F(x_i) = (\hat{y}, \hat{h}) = \underset{(y,h) \in \mathrm{GEN}(x_i)}{\operatorname{argmax}} w^T \phi(x_i, y, h) \quad (3)$$

which states that given ϕ and w, argmax returns the highest scoring translation y, maximizing over correspondences h (Och and Ney 2004). Following Liang et al. (2006), we assume that in our experiments every translation candidate is always coupled with a corresponding h, generated by (3). Thus, the following discussion on model training using SGD also applies to models with hidden variables if we define ϕ as in (3).

SGD has been widely used for discriminative training in NLP. The algorithm starts with an initial predictor w, and updates it for each training sample:

$$w_{k+1} = w_k - \eta \cdot g(w_k) \qquad (4)$$

where g is the subgradient with respect to a loss function $g(w) \in \partial \mathrm{loss}(w)$, and η the learning rate.

The loss functions that are commonly used in the N-best list based reranking tasks can be grouped into two categories. The first, including the hinge loss and logistic loss, takes into account only two candidates among $\mathrm{GEN}(x_i)$: the oracle candidate y^* in (2), and the highest scored incorrect candidate y', defined as

$$y' = \underset{y \in \mathrm{GEN}(x_i) \setminus \{y^*\}}{\operatorname{argmax}} w^T \phi(x_i, y).$$

Thus, under the reranking framework, the hinge loss is defined as

$$\begin{aligned} \mathrm{loss}_i(w) &= \max\{0, 1 - w^T(\phi(x_i, y^*) - \phi(x_i, y'))\} \\ &= \max\{0, 1 - w^T z_i\}. \end{aligned} \qquad (5)$$

If we define $\mathrm{loss}_i(w) = 0$ when the predicted y according to w, as in (1), is correct, then it is easy to

Inputs: training samples $\{(x_1, y_1^*), \ldots, (x_m, y_m^*)\}$, number of epochs T, $\eta > 0$, and $\lambda \geq 0$.
Initialization: $k = 1, w_1 = 0, c_1 = 0, \bar{g}_0 = 0$
Algorithm:
repeat
for $i = 1, \ldots, m$ **do**
$\hat{y} = \text{argmax}_{y \in \text{GEN}(x_i)} w_k^T \phi(x_i, y)$
if $\hat{y} = y_i^*$ **then**
$c_k = c_k + 1$
else
compute subgradient $g_k \in \partial \text{loss}(w_k)$
$\bar{g}_k = \big((k-1)\bar{g}_{k-1} + g_k\big)/k$
update w according to (9)
$c_k = 1$
$k = k + 1$
end if
end for
until T times
Output: total number of mistakes $M = k$, and $\{(w_1, c_1) \ldots (w_M, c_M)\}$

Figure 1. The VRDA algorithm (training).

Given: a list of parameter vectors $\{(w_1, c_1) \ldots (w_M, c_M)\}$
Input: x and its candidate set $\text{GEN}(x) = (y_1 \ldots y_J)$
Initialization: Set $v[j] = 0$ for $j = 1 \ldots J$ ($v[j]$ stores the number of votes for y_j)
Algorithm:
for $k = 1 \ldots M$ **do**
$y_j = \text{argmax}_{y_j \in \text{GEN}(x)} w_k^T \phi(x, y_j)$
$v[j] \mathrel{+}= c_k$
end for
Output: $F(x) = y_j$ where $j = \text{argmax}_j v[j]$

Figure 2. The VRDA algorithm (testing).

verify that to train a predictor using the hinge loss of (5), the update rule of (4) can be rewritten as

$$w_{k+1} = \begin{cases} w_k, & \text{if } \hat{y} = y^* \\ w_k + \eta z, & otherwise \end{cases}$$

Following Shalev-Shwartz (2012), by setting $\eta = 1$, we reach the well-known perceptron algorithm.

Similarly, the logistic loss is defined as $\text{loss}_i(w) = \log(1 + \exp(-w^T z_i))$.

The second category of loss functions takes into account the distribution over all candidates in an N-best list. Among them, log loss is widely used when a probabilistic interpretation of the trained model is desired, as in conditional random fields (Lafferty et al. 2001). Given a training sample, log loss is defined as $-\log P_w(y^*|x)$, where $P_w(y^*|x)$ is computed as

$$P_w(y^*|x_i) = \frac{\exp w^T \phi(x_i, y^*)}{\sum_{y \in \text{GEN}(x_i)} \exp w^T \phi(x_i, y)}. \quad (6)$$

One might also use a probability distribution of (6) and an application-specific cost function, as in (2), to define a loss which is more tightly coupled with the evaluation metric of the application. Such a loss function is sometimes called *Bayes risk* (Gimpel and Smith 2012), and is of the form

$$\text{loss}_{\text{B\_risk}} = \sum_{y \in \text{GEN}(x_i)} P_w(y|x_i) \text{cost}(y_i, y). \quad (7)$$

Unlike the other loss functions aforementioned, Bayes risk is non-convex, so there is no theoretical guarantee (e.g., convergence). However, recent empirical studies show its effectiveness in discriminative training for various tasks including speech recognition and SMT (Kaiser et al. 2000; Povey and Woodland 2002; Smith and Eisner 2006; Zens et al. 2007; Li and Eisner 2009; He and Deng 2012).

3 The Voted Regularized Dual Averaging Method

This section describes the VRDA algorithm, compares it with related work, and gives some formal properties of the algorithm.

3.1 The Algorithm

Consider a linear model w, in which predictions are made according (1) or (3). The goal of model training is to learn w with small expected loss on unseen data. To achieve this goal, VRDA trains the linear model by solving a problem of the form

$$w^* = \underset{w}{\text{argmin}} \left(\Psi(w) + \frac{1}{m} \sum_{i=1}^{m} \text{loss}_i(w) \right), \quad (8)$$

where $\Psi(w)$ is a convex regularization function.

More specifically, the VRDA algorithm is described in Figures 1 and 2, for training and testing respectively. The training module (Figure 1) takes T passes (epochs) over the training set, and only updates the parameter vector w_k when it makes a mistake, i.e., the highest scored candidate under w_k is not the oracle one. Each w_k is associated with a counter c_k, which counts the number of samples it processed correctly. These counts are then used in the testing module (Figure 2) as the voting weights to generate a prediction on a test sample.

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

$$w_{k+1} = \operatorname*{argmin}_w \left\{ \bar{g}_k^T w + \Psi(w) + \frac{\eta}{\sqrt{k}} \Omega(w) \right\}, \quad (9)$$

where $\Omega(w)$ is an auxiliary strongly convex function (as known as the *proximal function*), $\eta > 0$ is a parameter that controls the learning rate, and $\bar{g}_k$ is the average of subgradients g over k samples with prediction mistakes

$$\bar{g}_k = \frac{1}{k} \sum_{m=1}^{k} g_m.$$

For L_1 regularization used in this study, we define

$$\Psi(w) = \lambda \|w\|_1 \text{ and } \Omega(w) = \frac{1}{2} \|w\|_2^2.$$

Thus, the update rule (9) has a closed-form solution that employs the shrinkage (soft-thresholding) operator:

$$w_{k+1} = -\frac{\sqrt{k}}{\eta} \text{shrink}(\bar{g}_k, \lambda). \quad (10)$$

For a given g and a given truncation threshold $\lambda \geq 0$ that controls the sparsity of w, the shrinkage operator is defined coordinate-wise as

$$\left(\text{shrink}(g, \lambda)\right)^{(i)} = \begin{cases} g^{(i)} - \lambda, & \text{if } g^{(i)} > \lambda, \\ 0, & \text{if } \left|g^{(i)}\right| \leq \lambda, \\ g^{(i)} + \lambda, & \text{if } g^{(i)} < \lambda, \end{cases}$$

for $i = 1 \ldots d$.

The update rule (9) is a variant of the dual averaging (DA) method (Nesterov 2009) for effectively handling problems with simple regularization functions. Conceptually, the DA method in the batch mode is related to *cutting-plane* methods, where gradients from all previous iterations are used to construct a polyhedral lower bound model for the objective function, i.e., each gradient contributes a supporting hyperplane. However, such a model is very expensive to store and manipulate. The DA method effectively reduces the model to a single hyperplane lower bound (no longer supporting) by using the average gradient ($\bar{g}_k$), with an additional quadratic term for regularization ($\Omega(w)$). The algorithm name *dual averaging* comes from the fact that the gradients live in the dual space of $\{w\}$.

3.2 Comparisons with Related Work

This section elaborates the differences between the RDA update rule (9) and several related work. We note that the algorithms used in Martins et al. (2011a; 2011b) and Tsuruoka et al. (2009) are similar to the truncated stochastic gradient methods of Langford et al. (2009) and Duchi and Singer (2009). These algorithms can be considered as variants of the basic form

$$w_{k+1} = \operatorname*{argmin}_w \left\{ g_k^T w + \Psi(w) + \frac{\|w - w_k\|_2^2}{2\alpha_k} \right\} \quad (11)$$

Compared with this form, the RDA method (9) uses the average subgradient $\bar{g}_k$ instead of the current subgradient g_k; it uses a global proximal function say $\Omega(w) = (1/2)\|w\|_2^2$, instead of its local Bregman divergence $(1/2)\|w - w_k\|_2^2$; moreover, the coefficient for the proximal function is $\eta/\sqrt{k}$ instead of $1/\alpha_k = \sqrt{k}/\eta'$ for some constant η', where $\alpha_k = \eta'/\sqrt{k}$ is the step size. Although these two types of methods have the same order of iteration complexity, the three differences listed above contribute to quite different properties of their solutions. More specifically, the solution to (11) takes a form similar to (10):

$$w_{k+1} = \text{shrink}(w_k - \alpha_k g_k, \alpha_k \lambda). \quad (12)$$

It is clear that when k is large, the truncation threshold $\alpha_k \lambda$ is much smaller than λ, which is used in the RDA update. This causes the solution (12) much less sparse than the RDA update (10). We note that the lazy update or cumulative penalty scheme suggested by Langford et al. (2009) and Tsuruoka et al. (2009) aims to increase the truncation threshold in (12), but is still less effective than the RDA update, as we will demonstrate empirically in our experiments described in Sections 5 and 6 of this paper.

To better understand the connection and difference between the VRDA method and the truncated gradient method, we consider the special case $w_0 = 0$ and $\Psi(w) = 0$ for all w. In this case, the RDA update becomes

$$w_{k+1} = \frac{1}{\eta\sqrt{k}} \sum_{m=1}^{k} g_m, \quad (13)$$

and the update of the truncated gradient method becomes

$$w_{k+1} = \sum_{m=1}^{k} \frac{\eta'}{\sqrt{m}} g_m. \quad (14)$$

We note that (14) is just the traditional SGD method with diminishing step size, i.e., new gradients enter

the update with decreasing weights. In (13), however, all gradients have the same weight, which is scaled down at each iteration.

3.3 Formal Properties

This section gives some formal properties of the VRDA algorithm, using the well-studied voted perceptron algorithm (Freund and Schapire 1999; Collins 2002a) as a reference for comparison. We consider the following three questions:

(1) Does VRDA training (Figure 1) without regularization achieve the same mistake bound on training data as voted perceptron?
(2) How does the additional L_1 regularization term affect the mistake bound?
(3) How well does VRDA generalize to unseen test samples (Figure 2)?

We answer these questions by presenting two important theorems. We start with some definitions. Let *M* be the number of mistakes made by VRDA training (Figure 1) after processing m training samples. Recall that VRDA updates *w* only when it makes a mistake. We thus use $i(k)$ to denote the index of the sample on which the *k*-th mistake was made by w_k.

Let u be an (unknown) optimal predictor. We define the *total loss of* u over the subsequence $\{i(k)\}_{k=1}^{M}$, denoted by $L(u)$, as

$$L(u) = \sum_{k=1}^{M} \text{loss}_{i(k)}(u). \qquad (15)$$

We also define the *relative strength of regularization* of a sequence of learned vectors w_k with respect to u, denoted by $\Delta(u)$, as

$$\Delta(u) = \Psi(u) - \frac{1}{M}\sum_{k=1}^{M} \Psi(w_k). \qquad (16)$$

We then have Theorem 1 (see Appendix for the proof):

Theorem 1 *Let* (x_i, y_i) *for* $i = 1 \ldots m$ *be a sequence of labeled training data such that* $\forall i, \forall y \in \text{GEN}(x_i)\backslash\{y^*\}$, $\| \phi(x_i, y^*) - \phi(x_i, y)\|_2 \le R$, *where* y^* *is defined in (2). For any vector* u, *let* $L(u)$ *and* $\Delta(u)$ *be defined in (15) and (16), respectively. For the first pass over the training data of the VRDA algorithm in Figure 1 with* $\lambda\Delta(u) < 1$, *the number of mistakes M is bounded by*

$$M \le \inf_{u:1-\lambda\Delta(u)>0} \left(\sqrt{\frac{L(u)}{1-\lambda\Delta(u)}} + \frac{\sqrt{2}R\|u\|_2}{1-\lambda\Delta(u)} \right)^2. \qquad (17)$$

To answer the first question, we set $\lambda = 0$ (the case without regularization) and rewrite (17) as

$$M \le \inf_{u} \left(\sqrt{L(u)} + \frac{\sqrt{2}R}{\gamma} \right)^2, \gamma = \frac{1}{\|u\|_2} \qquad (18)$$

which bears a strong resemblance to the mistake bound of the voted perceptron algorithm (i.e., Theorem 2 in Collins (2002a)). The inequality (18) implies that if there exists some u such that $L(u)$ is relatively small, then the algorithm will make a small number of mistakes. Thus, Theorem 1 also shows that the VRDA algorithm can be robust to some training samples where the oracle candidate cannot be distinguished easily, if not impossible, from the rest of the candidates in $\text{GEN}(x)$, which are very common in NLP tasks.

Now, consider a special case where hinge loss (5) is used for VRDA and the training data is separable as stated in Assumption 1.

Assumption 1 *A training sequence* (x_i, y_i) *for* $i = 1 \ldots m$ *is said to be* ***separable with margin*** γ, *if there exists a vector* u *with* $\|u\|_2 = 1/\gamma$ *such that*

$$\forall i, \forall y \in \text{GEN}(x_i)\backslash\{y^*\}, u^T(\phi(x_i, y^*) - \phi(x_i, y)) \ge 1$$

where y^* *is the oracle candidate defined in (2).*

This assumption implies that $L(u)$ in (18) has a zero value for separable data. (18) becomes

$$M \le 2\left(\frac{R}{\gamma}\right)^2, \qquad (19)$$

which matches the mistake bound on separable data for the voted perceptron algorithm (i.e., Theorem 1 in Collins (2002a)), with an extra factor of two. (19) implies that if there exists a parameter vector that can make zero error on the training set, then after a finite number of iterations the training algorithm will have converged to a parameter vector with zero training error. We also notice that the mistake bound is independent of the dimension d (i.e., the number of features) and the number of candidates for each sample (i.e., the size of $\text{GEN}(x_i)$ for each input x_i). This is an important property because in many NLP tasks, such as parsing and SMT, the number of features could amount to tens of millions and $\text{GEN}(x_i)$ can be exponential in the size of the inputs.

Now, we give the answer to the second question. In the case of regularization i.e., $\lambda > 0$ in (17), the

mistake bound also depends on the relative strength of regularization $\Delta(u)$, defined in (16), which is the difference between $\Psi(u)$ and the average of the predictors generated by VRDA, $\Psi(w_1) \ldots \Psi(w_M)$. 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 three scenarios:

The optimal regularization case: $\Delta(u) = 0$. This happens if the optimal value of λ is chosen. In this case, we reach the same mistake bound as the case without regularization, as shown in (18).

The under-regularization case: $\Delta(u) < 0$. This happens if the value of λ is chosen too small, and the generated vectors $w_1 \ldots w_M$ on average has a larger Ψ value than $\Psi(u)$. In this case, we have a smaller mistake bound than the case of optimal regularization (when $\Delta(u) = 0$). This effect may be related to over-fitting on the training set.

The over-regularization case: $\Delta(u) > 0$ and $\lambda|\Delta(u)| < 1$. This happens if the value of λ is chosen too large, and the generated vectors $w_1 \ldots w_M$ on average has a smaller value of Ψ than $\Psi(u)$. In this case, the mistake bound can be much larger than the case of optimal regularization (when $\Delta(u) = 0$). If $\lambda\Delta(u) \geq 1$, (17) does not give any meaningful mistake bound (see Appendix for the proof).

To answer the third question, i.e., how well does VRDA generalize to unseen test samples, we give Theorem 2. Following Collins (2002a), we assume that there is some unknown distribution $P(x, y)$ over the set $X \times Y$, and that both training and test samples are drawn i.i.d. from this distribution. Our result (Theorem 2) is a direct corollary of Theorem 3 of Freund and Schapire (1999), which is a result of the theory developed in Helmbold and Warmuth (1995).

Theorem 2 *Assume all samples are generated i.i.d. at random. Suppose that we run the training algorithm in Figure 1 on a sequence of samples* $\{(x_1, y_1) \ldots (x_{m+1}, y_{m+1})\}$ *and M mistakes occur on samples with indices* $i(1) \ldots i(M)$. *Let* $L(u)$ *and* $\Delta(u)$ *be defined in (15) and (16), respectively.*

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

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

(The above expectation $E(.)$ *is over the choice of all* $m + 1$ *random samples.)*

Theorem 2 implies that if the VRDA algorithm makes a relatively small number of mistakes on training samples then it is likely to generalize well to unseen test samples.

# Algorithm	F-score	Prec.	Recall	NNZ
1. Baseline	0.8986	0.8983	0.8990	n/a
2. Perceptron	0.9164^{α}	0.9191	0.9143	950K
3. TG (hinge)	0.9172^{α}	0.9198	0.9127	775K
4. TG (logistic)	0.9165^{α}	0.9190	0.9139	485K
5. VRDA (hinge)	0.9176^{α}	0.9200	0.9191	542K
6. VRDA(logistic)	0.9179^{α}	0.9205	0.9153	902K

Table 1. Performance (on the test set) of different algorithms. NNZ stands for *the number of non-zero weights* in the model. The superscript α indicates statistically significant difference $p < 0.05$ from **Baseline**.

λ	hinge loss		logistic loss	
	F-score	NNZ	F-score	NNZ
0	0.9171	945K	0.9176	936K
1E-5	0.9175	900K	0.9179	902K
2E-5	0.9175	860K	0.9180	857K
5E-5	0.9166	713K	0.9165	713K
1E-4	0.9176	542K	0.9171	542K
2E-4	0.9165	374K	0.9170	373K
5E-4	0.9135^{α}	210K	0.9138^{α}	209K

Table 2. F-scores (on the test set) vs. the sparsity (measured by NNZ) of models trained using VRDA with different values of λ. The superscript α indicates statistically significant difference $p < 0.05$ from the models trained without regularization.

4 Parse Reranking

We follow the experimental paradigm outlined in Charniak and Johnson (2005). We used the same generative baseline model for generating candidate parses, and the 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, used Section 20-21 to optimize training parameters, such as the regularization parameters, the learning rate η and the number of iterations, and then evaluated the predictors on Section 22. The training set contains 36K sentences, while

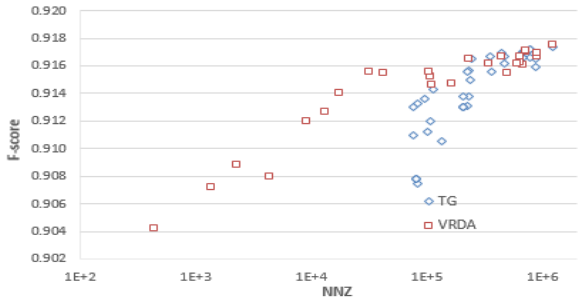

Figure 3. F-scores (on the test set) vs. NNZ of the models trained using hinge loss with TG and VRDA.

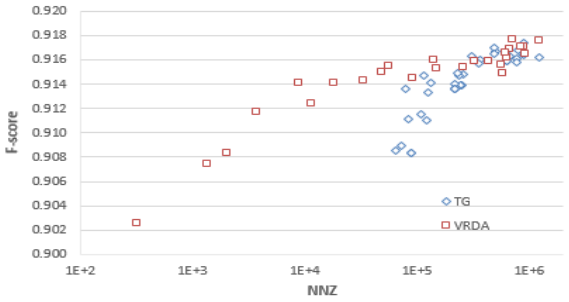

Figure 4. F-scores (on the test set) vs. NNZ of the models trained using logistic loss with TG and VRDA.

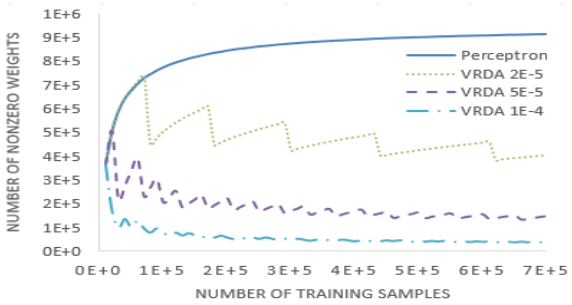

Figure 5. NNZ during the course of training using **Perceptron** and **VRDA** with hinge loss and different λ's.

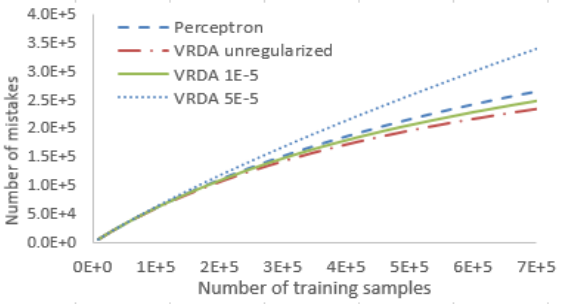

Figure 6. Number of mistakes on the training set during the course of training using **Perceptron** and **VRDA** with hinge loss and different λ's.

the development set and the test set have 4K and 1.7K, respectively. Performance of parsing re-ranking is measured with the PARSEVAL metric, i.e., F-score over labelled brackets.

Our main results are summarized in Table 1. **Baseline** (Row 1) is the parser of Charinak (2000). **Perceptron** (Row 2) is our implementation of the averaged perceptron algorithm (Collins 2002a). **TG** (Rows 3 and 4) is the truncated gradient method of (11) and (12). Our implementation of **TG** follows Langford et al. (2009). The truncation (12) is performed every N rounds. That is, if i/N is not an integer, we set λ in (12) to zero; otherwise, we let $\lambda = N\lambda'$ for a fixed gravity parameter $\lambda' > 0$. For **TG** and the **VRDA** algorithm (Rows 5 and 6), we trained linear models using the hinge loss and the logistic loss. For each type of loss, we report the result of the model which was trained using the parameter setting (including the number of iterations, the learning rate and the regularization parameters) optimized for F-score on development data. The results show that all the discriminatively trained models (Rows 2 to 6) significantly improve **Baseline**. Compared to **Perceptron**, **VRDA** and **TG** achieve slightly better F-scores with sparser models, measured in number of nonzero weights (NNZ).

Figures 3 and 4 compare the effectiveness of obtaining sparse solutions of **VRDA** and **TG**. For a fair comparison, we built a large number of linear models for each algorithm using different settings of training parameters that control the sparsity of the trained model. In **VRDA**, the model sparsity is controlled by the value of λ in (10). In **TG**, the model sparsity is controlled by N, the gravity parameter λ' and α_k in (12). The results show that in the cases where sparse models are desired **VRDA** is a better choice than **TG** since VRDA can produce much sparser models with less loss in F-score.

Table 2 and Figures 5 and 6 provide additional results to investigate in more detail the properties of **VRDA**. Table 2 presents the models trained with different values of λ. The results show that compared to the models trained without regularization, a better or very similar (with no statistically significant difference) F-score can be achieved by a much sparser model. This demonstrates the desired feature selection effect of **VRDA**, which helps prevent over-fitting. Figure 5 examines how NNZ changes during the course of training. As expected, larger λ's lead to sparser models whose training converges more quickly. In Figure 6, we plot the number of mistakes as a function of the number of training

samples received by **VRDA**. The results provide empirical justification of the analysis on mistake bounds presented in Section 3.3. First, we observed that the number of training errors grows sub-linearly with the number of training samples. Second, as predicted by Theorem 1, **VRDA** without regularization ($\lambda = 0$) makes no more training errors than **Perceptron**, but the number of training error increases along with more regularization ($\lambda > 0$).

5 Discriminative SMT

This section describes the use of VRDA for discriminative training of phrase translation models in SMT.

The phrase translation model, also known as the *phrase table*, consists of a list of bilingual phrase pairs. Each phrase pair is assigned with a translation score which in traditional phrase translation models is estimated based on counting the phrases, or their words, on an automatically word-aligned training data. In this study we have developed a new phrase translation model where the translation score for a phrase pair is learned using discriminative training methods based on SGD or VRDA. We refer to the new model as *discriminative phrase translation model* (DPTM). Formally, DPTM defines the translation score of a source-target sentence pair as

$$\begin{aligned} \text{score}_w(x_i, y, h) &= \sum_{i=1}^{d} w^{(i)}\, \phi^{(i)}(x, y, h) \\ &= w^T \phi(x, y, h), \end{aligned} \quad (20)$$

where d is the total number of phrase pairs in the phrase table, w is the weight vector to be learned, and $\phi^{(i)}(x, y, h)$ is an indicator function whose value is 1 if the i-th bilingual phrase is in h, and 0 otherwise. Recall that h is a hidden variable consisting of a segmentation of x and y into phrases and an alignment between source and target phrases, as in (3).

The parameters of the DPTM w can be learned using SGD or VRDA, together with a loss function. In addition to the classical functions, including hinge loss, logistic loss and log loss, as described in Section 2, we also used a Bayes risk function, which is based on the expected BLEU defined on N-best lists (e.g., Gao and He 2013). Given the current model w, Bayes risk over one training sample (i.e., x_i and its labeled N-best list) is defined as

$$\begin{aligned} &\text{loss}_{\text{B\_risk}}(w) \\ &= -\sum_{y \in \text{GEN}(x_i)} P_w(y|x_i)\, \text{sBleu}(y_i, y). \end{aligned} \quad (21)$$

(21) is a special case of (7) in that the procedure used to generate the N-best translation candidates GEN is a baseline phrase-based SMT system, which in our study is a reimplementation of the Moses system (Koehn et al. 2007) that does not use the DPTM, and the application-specific cost function is defined as the negative of the sentence-level BLEU score (He and Deng 2012), denoted by sBleu, which measures the quality of translation candidate y with respect to its reference translation y_i. $P_w(y|x_i)$ in (21) is a normalized translation probability from x_i to y computed using *softmax* as

$$P_{(w)}(y|x_i) = \frac{\exp\big(\text{score}_w(x_i, y, h) + \text{b}(x_i, y, h)\big)}{\sum_{y' \in \text{GEN}(x_i)} \exp\big(\text{score}_w(x_i, y', h) + \text{b}(x_i, y', h)\big)}$$

where $\text{score}_w(.)$ is defined by the DPTM in (20), and $\text{b}(.)$ is the baseline score produced by the baseline phrase-based SMT system. Since the DPTM has to work together with other component models of the SMT system, including the baseline score in the loss function forces the learning algorithm to estimate the parameters of the DPTM in such a way that the quality of end-to-end machine translation results is directly optimized.

The subgradient g of this Bayes risk is

$$g(w) = \sum_{y \in \text{GEN}(x_i)} \text{U}(w, y) P_w(y|x_i) \phi(x_i, y, h),$$

where $\text{U}(w, y) = \text{loss}_{\text{B\_risk}}(w) - \text{sBleu}(y_i, y)$.

Given the loss function and its subgradient, the parameters of the DPTM can be optimized using the SGD algorithm or the VRDA algorithm (Figure 1). After the DPTM is trained, we incorporated it as an additional feature into the log-linear model of SMT (3), where the feature weights are optimized using MERT (Och 2003) to maximize the BLEU score on development data.

5.1 Experiments

We conducted our experiments on the German-to-English (DE-EN) Europarl translation task (Koehn and Monz 2006). The training set contains 751K sentence pairs, with 21 words per sentence on average. The official development set used for the shared task contains 2000 sentences. In our experi-

	SGD		VRDA	
	TEST1	**TEST2**	**TEST1**	**TEST2**
Baseline	26.0	26.0	26.0	26.0
B_risk	26.8^{α}	26.7^{α}	26.9^{α}	26.7^{α}
hinge loss	26.4	26.2	26.5^{α}	$26.4^{\alpha\beta}$
logistic loss	26.2	26.3	$26.6^{\alpha\beta}$	26.4
log loss	26.6^{α}	26.4	26.6^{α}	$26.6^{\alpha\beta}$

Table 3. BLEU scores using the DPTMs trained with different algorithms. The superscripts α and β indicate statistically significant difference $p < 0.05$ from Baseline and SGD with the same loss function, respectively.

λ	**NNZ**	**TEST1**	**TEST2**
0	2.5M	26.9	26.7
2E-8	357K	26.9	26.6
5E-8	154K	26.7	26.7
1E-7	57K	26.6	26.6
5E-7	5K	26.3^{α}	26.5^{α}

Table 4. BLEU scores vs. the sparsity (NNZ) of the DPTMs which are trained using VRDA with different values of λ for regularization. The superscript α indicates statistically significant difference $p < 0.05$ from the models trained without regularization.

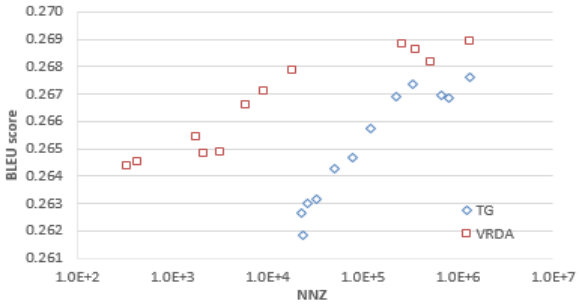

Figure 7. BLEU scores (on TEST1) vs. NNZ weights of the models trained using **B_risk** with VRDA and TG.

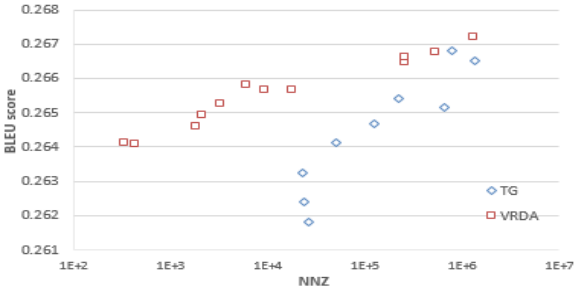

Figure 8. BLEU scores (on TEST2) vs. NNZ weights of the models trained using **B_risk** with VRDA and TG.

ments we used the first 1000 sentences as a development set for MERT training and optimizing parameters for discriminative training, such as learning rate and the number of iterations. We used the rest 1000 sentences as the first test set (TEST1). We used the WMT06 test data as the second test set (TEST2), which contains 2000 sentences.

The metric used for evaluation is case insensitive BLEU score (Papineni et al. 2002). We also performed a significance test using the paired *t*-test. Differences are considered statistically significant when the *p*-value is less than 0.05.

The main results are presented in Table 3. The baseline phrase-based SMT system is the same that we used for generating the N-best lists for discriminative training. Results show that the effectiveness of discriminative training, using either SGD or VRDA, depends to a large degree upon the choice of loss functions. The loss functions that take into account the distribution over all hypotheses in an N-best list (i.e., Bayes risk and log loss) are more effective than the ones that do not. Bayes risk, despite its non-convexity, significantly outperforms the others because it combines the cost function (i.e., sBleu) that is closely coupled with the evaluation metric under consideration (i.e., BLEU). Overall, VRDA compares favorably to SGD. In addition, VRDA shows robust performance for both the convex loss and the non-convex loss (Bayes risk). In the former case, it outperforms SGD with a statistically significant margin in some runs. More importantly, as shown in Table 4, VRDA is effective in obtaining sparse models due to the use of the L_1 regularization term. For example, we can have a model that is an order of magnitude smaller, with negligible performance difference to the baseline. This is important for system deployment in practice.

Figures 7 and 8 compare VRDA to TG. The results are similar to that of the parse reranking results in Figures 3 and 4. Compared to TG, VRDA can produce much sparser models with less BLEU score loss.

6 Conclusion

Online methods are increasingly attractive for large-scale machine learning in recent studies (e.g., Zhang 2004; Bottou 2010). However, traditional methods cannot effectively induce particular structure (such as sparsity) into solutions. This paper presents the VRDA algorithm for large-scale discriminative training in NLP, as an alternative to the perceptron algorithms or SGD. The new algorithm estimates

model parameters by minimizing L_1 regularized objectives and are effective in obtaining sparse solutions. We give the mistake bound of the algorithm, and show how the bound is affected by the additional regularization term. Evaluations are performed on two NLP tasks: parse reranking and discriminative phrase translation model training for SMT, showing that VRDA gives better and sparser solutions than perceptron or SGD, and that VRDA generates substantially sparser solutions than the truncated gradient method, which is a state-of-the-art method of sparse learning for linear models.

References

Andrew, G., and Gao, J. 2007. Scalable training of l1-regularized log-linear models. In *ICML*, 33-47.

Bottou, L. O. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT, 177-187*.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *NAACL*, 132-139.

Charniak, E., and Johnson, M. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*, 173-180.

Collins, M. 2000. Discriminative re-ranking for natural language parsing. In *ICML*, 175-182.

Collins, M. 2002a. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP*, 1-8.

Collins, M. 2002b. New ranking algorithms for parsing and tagging: kernels over discrete structures and the voted perceptron. In *ACL*.

Duchi, J. and Singer, Y. 2009. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2873-2908.

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

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

Gao, J., Andrew, G., Johnson, M., and Toutanova, K. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *ACL*, 824-831.

Gao, J., and He, X. 2013. Training MRF-based translation models using gradient ascent. In *NAACL-HLT*, pp. 450-459.

Gimpel, K., and Smith, N. A. 2012. Structured ramp loss minimization for machine translation. In *NAACL-HLT*.

He, X., and Deng, L. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *ACL*, pp. 292-301.

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

Kaiser, J., Horvat, B., and Kacic, Z. 2000. A novel loss function for the overall risk criterion based discriminative training of hmm models. In *ICSLP*.

Koehn, P., and Monz, C. 2006. Manual and automatic evaluation of machine translation between European languages. In *Workshop on Statistical Machine Translation*, pp. 102-121.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. 2007. Moses: open source toolkit for statistical machine translation. In *ACL 2007*, demonstration session.

Lafferty, J., McCallum, A., and Pereira, F. 2001. Conditional random fields: probablistic models for segmenting and labeling sequence data. In *ICML*.

Li, Z., and Eisner, J. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *ACL*.

Langford, J., Li, L., and Zhang, T. 2009. Sparse online learning via truncated gradient. *JMLR*, 10:777-801.

Liang, P., Bouchard-Cote, A. Klein, D., and Taskar, B. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL*.

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. 2011a. Online

learning of structured predictors with multiple kernels. In *AISTATS*.

Martins, A. F. T., Smith, N. A., T., Aguiar, P. M. Q., and Figueiredo, M. A. 2011b. Structured sparsity in structured prediction. In *EMNLP*.

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

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 29(1): 19-51.

Och, F. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pp. 160-167.

Papineni, K., Roukos, S., Ward, T., and Zhu W-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Povey, D. and Woodland, P. C. 2002. Minimum phone error and i-smoothing for improved discriminative training. In *ICASSP*.

Roark, R., Saraclar, M., and Collins, M. 2007. Discriminative n-gram language modeling. Computer Speech and Language, 21(2):373-392.

Shalev-Shwartz, Shai. 2012. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107-194.

Shen, L., Sarkar, A., and Och, F. 2004. Discriminative reranking for machine translation. In *HLT/NAACL*.

Simianer, P., Riezler, S., and Dyer, C. 2012. Joint feature selection in distributed stochasic learning for large-scale discriminative training in SMT. In *ACL*, pp. 11-21.

Smith, D. A., and Eisner, J. 2006. Minimum risk annealing for training log-linear models. In *COLING-ACL*.

Tsuruoka, T., Tsujii, J., and Ananiadou, S. 2009. Stochastic gradient descent training using l1-regularized log-linear models with cumulative penalty. In *ACL*.

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

Zen, R., Hasan, S., and Ney, H. 2007. A systematic comparsion of training criteria for statistical machine translation. In *EMNLP*.

Zhang, T. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, 116-123.

Appendix A. The Proof of Theorem 1

This section gives the proof of Theorem 1 by providing an analysis of the voted regularized dual averaging (VRDA) algorithm for the case $T = 1$ (i.e., going through the training set once). The analysis parallels that for the voted perceptron algorithm given in Freund and Schapire (1999).

We bound the number of mistakes made by VRDA through its regret analysis. First, we recognize that VRDA is equivalent to running RDA (Xiao 2010) on the subsequence of training samples where a prediction mistake is made. Let M be the number of mistakes made by the algorithm after processing m training samples, and $i(k)$ denote the index of the sample on which the k-th mistake was made (by w_k). The regret of the algorithm, with respect to a fixed vector u, is defined only by the samples with prediction errors:

$$R_M(u) = \sum_{k=1\ldots M}\left(\text{loss}_{i(k)}(w_k) + \Psi(w_k)\right) - \sum_{k=1\ldots M}\left(\text{loss}_{i(k)}(u) + \Psi(u)\right). \quad \text{(A.1)}$$

According to Theorem 1 and Corollary 2 of Xiao (2010), by setting $\eta = \sqrt{2}G/\|u\|_2$, the RDA method has the regret bound as

$$R_M(u) \le \sqrt{2}G\|u\|_2\sqrt{M} \quad \text{(A.2)}$$

where G is an upper bound on the norm of the sub-gradients, i.e., $\|g_k\|_2 \le G$ for all $k = 1 \ldots M$. Since the loss functions are upper bounds for the 0-1 loss

$$M \le \sum_{k=1\ldots M}\text{loss}_{i(k)}(w_k),$$

we can bound the number of mistakes by combining the above inequality with (A.1) and (A.2) as

$$M \le L(u) + M\lambda\Delta(u) + \sqrt{2}G\|u\|_2\sqrt{M} \quad \text{(A.3)}$$

where $L(u)$ is the *total loss* of the vector u over the subsequence $\{i(k)\}_{k=1}^{M}$, defined as

$$L(u) = \sum_{k=1}^{M}\text{loss}_{i(k)}(u), \quad \text{(A.4)}$$

and $\Delta(u)$ is the *relative strength of regularization* of w_k with respect to u, defined as

$$\Delta(u) = \Psi(u) - \frac{1}{M}\sum_{k=1}^{M}\Psi(w_k). \quad \text{(A.5)}$$

A.1 Analysis for separable data

Our analysis for separable data is based on the hinge loss defined in (5) of the paper, and Assumption 1 described in Section 3.3 of the paper. Assumption 1 is adapted from the standard *separability with margin* assumption. Under this assumption, we have $L(u) = 0$ in (A.3) and the margin of separability is defined as $\gamma = 1/\|u\|_2$. For convenience, we let

$$R = \max_{i=1\ldots N}\|z_i\|_2.$$

Then we can set $G = R$ since for hinge loss of (5), $-z_i$ is the subgradient of $\text{loss}_i(u)$, and we have $\|-z_i\|_2 = \|z_i\|_2 \le R$ 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

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

  which is very similar to the mistake bound for the voted perceptron algorithm (Freund and Schapire 1999), with an extra factor of two. Note that this bound is independent of the dimension d and the number of samples m. It also holds for $T > 1$ (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 average of $\Psi(w_1) \ldots \Psi(w_M)$. More specifically,

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

  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 value of λ is chosen too small, and the generated vectors $w_1 \ldots w_M$ on average has a larger Ψ value than $\Psi(u)$. In this case, we have

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

  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 appens if the value of λ is chosen too large, and the generated vectors $w_1 \ldots w_M$ on average has a

smaller value in Ψ than $\Psi(u)$. If in addition $\lambda|\Delta(u)| < 1$, then we have

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

which can be much larger than the case of *perfect* regularization (when $\Delta(u) = 0$). If $\lambda\Delta(u) \ge 1$, then the inequality of (A.3) holds trivially and does not give any meaningful mistake bound.

A.2 Analysis for inseparable data

Our analysis is similar to the error analysis for the perceptron in Shalev-Shwartz (2012), which relies on applying the following lemma to (A.3)

Lemma 1 *Given $a, b, c > 0$, the inequality $x - b\sqrt{x} - c \le 0$ implies*

$$x \le \frac{c}{a} + \left(\frac{b}{a}\right)^2 + \frac{a}{b}\sqrt{\frac{c}{a}} \le \left(\sqrt{\frac{c}{a}} + \frac{b}{a}\right)^2.$$

Here are the case-by-case analysis:

- if $\lambda = 0$, we have

  $$M \le L(u) + \sqrt{2}R\|u\|_2\sqrt{M}.$$

  which results in (using Lemma 1 with $a = 1$)

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

  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: $\Delta(u) < 0$. Using Lemma 1 with $a = 1 + \lambda|\Delta(u)|$, we have

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

  The over-regularization case: $\Delta(u) > 0$. If $\lambda|\Delta(u)| < 1$, then using Lemma 1 with $a = 1 - \lambda|\Delta(u)|$, we have

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

  Again, if $\lambda\Delta(u) \ge 1$, the inequality (A.3) holds trivially and does not lead to any meaningful bound.

In summary, we have proved the following theorem.

Theorem 1 *Let (x_i, y_i) for $i = 1 \ldots m$ be a sequence of labeled training data such that $\forall i, \forall y \in \mathrm{GEN}(x_i)\backslash\{y^*\}$, $\|\phi(x_i, y^*) - \phi(x_i, y)\|_2 \le R$. For any vector u, let $L(u)$ be the total loss defined in (A.4), and $\Delta(u)$ be the relative strength of regularization defined in (A.5). For the first pass over the training data of the VRDA algorithm in Figure 1 with $\lambda\Delta(u) < 1$, the number of mistakes M is bounded by*

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

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

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

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

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