Large Margin Winnow Methods for Text Categorization

Tong Zhang
Mathematical Sciences Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598 USA
tzhang@watson.ibm.com

ABSTRACT
The SNoW (Sparse Network of Winnows) architecture has recently been successful applied to a number of natural language processing (NLP) problems. In this paper, we propose large margin versions of the Winnow algorithms, which we argue can potentially enhance the performance of basic Winnows (and hence the SNoW architecture). We demonstrate that the resulting methods achieve performance comparable with support vector machines for text categorization applications. We also explain why both large margin Winnows and SVM can be suitable for NLP tasks.

1. INTRODUCTION
Recently there have been considerable interests in applying machine learning techniques to problems in natural language processing. One method that has great success in many applications is the SNoW architecture [5, 12]. This architecture is based on the Winnow algorithm [15], which in theory is suitable for problems with many irrelevant attributes. The success of SNoW is then attributed to the argument that typical NLP tasks are of very high dimension but most features are irrelevant. On the other hand, recently there have been many developments on large margin Perceptron algorithms (SVMs), leading to the state of the art performance on text categorization [11, 6]. In [11], Joachims argued that the success of SVM is due to many relevant features rather than irrelevant features for text categorization problems. Such relevant features can be picked up by an SVM.

It is therefore helpful to investigate this issue further so as to gain a better understanding on which approach is more suitable for NLP problems, especially for text categorization. Since SVM is a large margin version of the Perceptron, we shall thus compare it with large margin versions of the Winnow algorithms, which we derive later in the paper. Our study indicates that both Winnow family and Perceptron family of algorithms can achieve the same level of performance on text categorization. However, the Perceptron family is more sensitive to feature selection. We also provide a theoretical explanation that is consistent with this finding.

Before we proceed, we formalize the problem considered in this paper as binary classification: to determine a label $y \in \{-1, 1\}$ associated with an input vector x. Since we are concerned with linear classifiers, our task is to seek a weight vector w and a threshold θ such that $w^T x < \theta$ if its label $y = -1$ and $w^T x \geq \theta$ if its label $y = 1$. Furthermore, we shall assume $\theta = 0$ in this paper for simplicity. This restriction does not cause problems in practice since one can always append a constant feature to the input data x, which offset the effect of θ. The formulation with $\theta = 0$ can be more amenable to theoretical analysis. For an SVM, a fixed threshold also allows a simple Perceptron like numerical algorithm as described in chapter 12 of [18], as well as in [16] and [9]. Note that although more complex, a non-fixed θ does not introduce any fundamental difficulty.

The paper is organized as follows. In Section 2, we review the Perceptron and the Winnows. Based on the derivation of large margin Perceptrons (SVMs), we show how large margin Winnow methods can be obtained. In Section 3, we discuss learning aspects of Perceptron and Winnow families of algorithms, and suggest how to use the results to interpret suitability of different methods for different applications. Experimental results will be given in Section 4. Finally, we make some concluding remarks in Section 5.

2. SVM AND LARGE MARGIN WINNOW

2.1 Perceptron
We review the derivation of SVM from Perceptron, which serves as a reference for our derivation of large margin Winnows.

Given a training set of n labeled data $(x^1, y^1), \ldots, (x^n, y^n)$, the Perceptron algorithm updates the weight vector w by going through the training data repeatedly. The algorithm is online in the sense that the update depends only on one training vector at a time; it is also mistake driven in the sense that the weight vector is updated only when the algorithm is not able to correctly classify an example. For Perceptron, the update rule is additive: if the linear discriminant function misclassifies an input training vector x^i with true label y^i, then we update each component j of the weight vector w as:

$$w_j \leftarrow w_j + \eta x_j^i y^i,$$

where $\eta > 0$ is a parameter called learning rate. The initial weight vector can be taken as $w = 0$.

For linearly separable problems, let w be a weight that separates the in-class vectors from the out-of-class vectors in the training set. It is well known that the Perceptron algorithm computes a weight that correctly classifies all training data after at most M updates (a proof can be found in [20]) where

$$M = \|w\|_2^2 \max_i \|x^i\|_2^2 / (\min_i w^T x^i)^2.$$

The weight vector w_* that minimizes the right hand side of the bound is called the optimal hyperplane in [20] or the optimal stability hyperplane in [1, 13, 17]. This optimal hyperplane is the solution to the following quadratic programming problem:

$$\begin{aligned} &\min_w \frac{1}{2} w^2 \\ &\text{s.t.} \quad w^T x^i y^i \geq 1 \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

In reality, not every problem is linearly separable. For such problems, as being proposed in [3], one can introduce a slack variable ξ^i for each data point (x^i, y^i) $(i = 1, \ldots, n)$, and compute a weight vector $w_*(C)$ that solves

$$\begin{aligned} &\min_{w,\xi} \frac{1}{2} w^T w + C \sum_i \xi^i \\ &\text{s.t.} \quad w^T x^i y^i \geq 1 - \xi^i, \quad \xi^i \geq 0 \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

Where $C > 0$ is a given parameter [20].

It is known that when $C \to \infty$, then $\xi_i \to 0$ and $w_*(C)$ converges to the weight vector w_* of the optimal hyperplane. We can write down the KKT condition for the above optimization problem, and let α^i be the Lagrangian multiplier for $w^T x^i y^i \geq 1 - \xi^i$. After elimination of w and ξ, we obtain the following dual optimization problem of the dual variable α (see [20], chapter 10 for details):

$$\begin{aligned} &\max_\alpha \sum_i \alpha^i - \frac{1}{2} (\sum_i \alpha^i x^i y^i)^2 \\ &\text{s.t.} \quad \alpha^i \in [0, C] \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

The weight $w_*(C)$ is given by

$$w_*(C) = \sum_i \alpha^i x^i y^i$$

at the optimal solution. To solve this problem, one can use the following modification of the Perceptron update algorithm (see [16, 9] and chapter 12 of [18]): at each data (x^i, y^i), we fix all α_k with $k \neq i$, and update α_i to maximize the dual objective functional, which gives:

$$\alpha^i \to \max(\min(C, \alpha^i + \eta(1 - w^T x^i y^i)), 0),$$

where $w = \sum_i \alpha^i x^i y^i$. The learning rate η can be set as $\eta = 1/x^{iT} x^i$ which corresponds to the exact maximization of the dual objective functional. However, in practice, it is useful to use a smaller learning rate due to some statistical and numerical reasons, which we do not have space to elaborate.

2.2 Winnow

Similar to Perceptron, the (unnormalized) Winnow algorithm (with positive weight), is also online and mistake driven. However, the update rule is multiplicative instead of additive: if the linear discriminant function misclassifies an input training vector x^i with true label y^i, then we update each component j of the weight vector w as:

$$w_j \leftarrow w_j \exp(\eta x_j^i y^i),$$

where $\eta > 0$ is the learning rate parameter, and the initial weight vector can be taken as $w_j = \mu_j > 0$. The Winnow algorithm belongs to a general family of algorithms called exponentiated gradient descent with unnormalized weights (EGU) [14]. There can be a number of variants. One modification is to normalize the one-norm of the weight w so that $\sum_j w_j = W$, which leads to the normalized Winnow. Another variant is called balanced Winnow, which is equivalent to an embedding of the input space into a higher dimensional space as: $\tilde{x} = [x, -x]$. This modification allows the positive weight Winnow algorithm for the augmented input $\tilde{x}$ to have the effect of both positive and negative weights for the original input x. Due to this simple construction of balanced Winnow, we will only focus on positive weight Winnows (both normalized and unnormalized) in our derivation.

Similar to a Perceptron, for a problem that is linearly separable with a positive weight w, the Winnow algorithm computes a solution that correctly classifies all training data after at most M updates with

$$M = 2W(\sum_j w_j \ln \frac{w_j \|\mu\|_1}{\mu_j \|w\|_1}) \max_i \|x^i\|_\infty^2 / \delta^2,$$

where $0 < \delta \leq \min_i w^T x^i y^i$, $W \geq \|w\|_1$ and the learning rate is $\eta = \delta / (W \max_i \|x^i\|_\infty^2)$. The technique for deriving the above bound was developed in [7] (also see [15] for earlier results). The detailed proof of this specific bound can be found in [24] which employed techniques in [7]. Note that unlike the Perceptron mistake bound, the above bound is learning rate dependent. It also depends on the prior $\mu_j > 0$ which is the initial value of w.

For problems separable with positive weights, to obtain an optimal stability hyperplane associated with the Winnow mistake bound, we consider fixing $\|w\|_1$ such that $\|w\|_1 = W > 0$. It is then natural to define the optimal hyperplane as the (positive weight) solution to the following convex programming problem:

$$\begin{aligned} &\min_w \sum_j w_j \ln \frac{w_j}{e\mu_j} \\ &\text{s.t.} \quad w^T x^i y^i \geq 1 \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

Similar to the derivation of SVM, for non-separable problems, we can introduce a slack variable ξ^i for each data (x^i, y^i), and compute a weight vector $w_*(C)$ that solves

$$\begin{aligned} &\min_{w,\xi} \sum_j w_j \ln \frac{w_j}{e\mu_j} + C \sum_i \xi^i \\ &\text{s.t.} \quad w^T x^i y^i \geq 1 - \xi^i, \quad \xi^i \geq 0 \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

Where $C > 0$ is a given parameter. Note that to derive the above methods, we have assumed that $\|w\|_1$ is fixed at $\|w\|_1 = \|\mu\|_1 = W$, where W is a given parameter. This implies that the derived methods are in fact large margin versions of the normalized Winnow. However in practice, one can also ignore this normalization constraint so that the derived method corresponds to large margin versions of the unnormalized Winnow — note that a small relative entropy also implies a small 1-norm.

Similar to an SVM, the non-separable formulations of large margin Winnows approach the separable formulation as $C \to \infty$. We thus only focus on the non-separable case below. Also similar to an SVM, we can write down the KKT condition and let α^i be the Lagrangian multiplier for $w^T x^i y^i \geq 1 - \xi^i$. After elimination of w and ξ, we obtain (the algebra resembles that of [20], chapter 10) the following dual formulation for large margin unnormalized Winnow:

$$\max_\alpha \sum_i \alpha^i - \sum_j \mu_j \exp(\sum_i \alpha^i x^i_j y^i)$$
$$\text{s.t.} \quad \alpha^i \in [0, C] \quad \text{for } i = 1, \dots, n.$$

The j-th component of weight $w_*(C)$ is given by

$$w_*(C)_j = \mu_j \exp(\sum_i \alpha^i x^i_j y^i)$$

at the optimal solution. For large margin normalized Winnow with $\|w\|_1 = W > 0$, we obtain

$$\max_\alpha \sum_i \alpha^i - W \ln(\sum_j \mu_j \exp(\sum_i \alpha^i x^i_j y^i))$$
$$\text{s.t.} \quad \alpha^i \in [0, C] \quad \text{for } i = 1, \dots, n.$$

The weight $w_*(C)$ is given by

$$w_*(C)_j = W\mu_j \exp(\sum_i \alpha^i x^i_j y^i) / \sum_j \mu_j \exp(\sum_i \alpha^i x^i_j y^i)$$

at the optimal solution.

Similar to the Perceptron-like update rule for the dual SVM formulation, it is possible to derive Winnow-like update rules for the large margin Winnow formulations. At each data (x^i, y^i), we fix all α_k with $k \neq i$, and update α_i to maximize the dual objective functionals. We use a gradient ascent method with a learning rate η: $\alpha_i \to \alpha_i + \eta \frac{\partial}{\partial \alpha_i} L_D(\alpha_i)$, where we use L_D to denote the dual objective function to be maximized. η can be either fixed as a small number or computed by the Newton's method. It is not hard to verify that we obtain the following update rule for large margin unnormalized Winnow:

$$\alpha^i \to \max(\min(C, \alpha^i + \eta(1 - w^T x^i y^i), 0),$$

where

$$w_j = \mu_j \exp(\sum_i \alpha^i x^i_j y^i).$$

This gradient ascent on the dual variable gives an EGU rule as in [14]. Compared with the SVM dual update rule which is a soft-margin version of the Perceptron update rule, this method naturally corresponds to a soft-margin version of unnormalized Winnow update. Similarly, we obtain the following dual update rule for large margin normalized Winnow:

$$\alpha^i \to \max(\min(C, \alpha^i + \eta(1 - w^T x^i y^i), 0),$$

where

$$w_j = W\mu_j \exp(\sum_i \alpha^i x^i_j y^i) / \sum_j \mu_j \exp(\sum_i \alpha^i x^i_j y^i).$$

Again, this rule (which is an EG rule in [14]) can be naturally regarded as the soft-margin version of the normalized Winnow update.

The entropy regularization condition is natural to all exponentiated gradient methods [14], as can be observed from the theoretical results in [14]. The regularized normalized Winnow is closely related to the maximum entropy discrimination [10] (the two methods are almost identical for linearly separable problems). However, in the framework of maximum entropy discrimination, the Winnow connection is non-obvious. Note also that the SNoW architecture for NLP problems employs a heuristics for a margin version of unnormalized Winnow as described in [5, 8]. However, the algorithm was purely mistake driven without dual variables α^i (therefore the algorithm does not automatically compute an optimal stability hyperplane for the Winnow mistake bound). In addition, the formulation does not explicitly include a regularization parameter C, which in practice could be very helpful for non-separable problems typically observed in NLP tasks.

3. PROPERTIES OF SVM AND LARGE MARGIN WINNOW

In order to understand the suitability of an SVM or a large margin Winnow for a specific application, it is useful to analyze their ability to correctly predict the label of a future data based on a set of n training data. In machine learning, a popular framework to measure this prediction ability is through the generalization error analysis. We make the standard assumption that the data (x, y) is taken from an unknown distribution D. Given n random samples from D, we would like to know what is the probability that the resulting classifier will have a small classification error. This style of analysis is often called PAC analysis. For an SVM, many such results can be found in chapter 4 of [4] and references therein. We list a variant of Theorem 4.19 in [4]:

THEOREM 3.1. *If the data is 2-norm bounded as $\|x\|_2 \leq b$, then consider the family Γ of hyperplanes w such that $\|w\|_2 \leq a$. Denote by $err(w)$ the misclassification error of w with the true distribution. Then there is a constant C such that for any $\gamma > 0$, with probability $1 - \eta$ over n random samples, any $w \in \Gamma$ satisfies:*

$$err(w) \leq \frac{k_\gamma}{n} + \sqrt{\frac{C}{\gamma^2 n} a^2 b^2 \ln(\frac{nab}{\gamma} + 2) + \ln\frac{1}{\eta}},$$

where $k_\gamma = |\{i : w^T x^i y^i < \gamma\}|$ is the number of samples with margin less than γ.

One difference of this version compared with Theorem 4.19 of [4] is that we have adopted a covering number estimate in [24] which is slightly better than what was used in [4] by a $\log n$ factor. Also see [21] for some related results. Another difference is that we use a data-independent margin γ rather than a data dependent margin as in [4]. These differences are relatively small. The bound is for linearly non-separable problems which are typical in real applications such as NLP tasks. For linearly separable problems, a better bound (without the square-root) can be obtained. The above theorem justifies the 2-norm regularization term used in an SVM when the data is 2-norm bounded. Similarly, we can obtain a theorem that justifies the entropy regularization (when the data is infinity-norm bounded) for large margin Winnows:

THEOREM 3.2. *If the data is infinity-norm bounded as* $\|x\|_\infty \le b$, *then consider the family* Γ *of hyperplanes* w *such that* $\|w\|_1 \le a$ *and* $\sum_j w_j \ln(\frac{w_j \|\mu\|_1}{\mu_j \|w\|_1}) \le c$. *Denote by* $err(w)$ *the misclassification error of* w *with the true distribution. Then there is a constant* C *such that for any* $\gamma > 0$, *with probability* $1-\eta$ *over* n *random samples, any* $w \in \Gamma$ *satisfies:*

$$err(w) \le \frac{k_\gamma}{n} + \sqrt{\frac{C}{\gamma^2 n} b^2 (a^2 + ac) \ln(\frac{nab}{\gamma} + 2) + \ln \frac{1}{\eta}},$$

where $k_\gamma = |\{i : w^T x^i y^i < \gamma\}|$ *is the number of samples with margin less than* γ.

This bound is a direct consequence of Theorem 2.2 and Theorem 2.8 in [24]. γ can also be made data-dependent as in [19] or [4].

Note that from the theoretical results, the main difference of an SVM and a large margin Winnow is the data assumption: if the data is 2-norm bounded and there is a small 2-norm hyperplane that achieves a large margin, then SVM is suitable for the problem; if the data is infinity-norm bounded and there is a hyperplane with a small 1-norm and a small entropy with respect to the prior μ that achieves a large margin, then Winnow is suitable for the problem.

The usual claim that if there are many irrelevant features, then the Winnow family of algorithms learn quickly can be explained by the above bounds. The underlying assumption for this claim is that the data is infinity-norm bounded;[1] and there is a small one norm hyperplane that can achieve a good margin.[2]

From this analysis, the advantage of the Winnow family of algorithms (compared with the Perceptron family of algorithms) requires the data to have small infinity-norm but large 2-norm. This phenomenon has been confirmed in [14] by numerical simulations. Specifically, they have shown that if the data are non-sparse and are infinity-norm bounded (in this case, the 2-norm of a data point is much larger than its infinity-norm), and the target function is sparse, then the Winnow family of algorithms learn more quickly. However, they have also shown that if the data is drawn from a unit sphere (so that the two-norm of the data is comparable with the infinity-norm), and the target function is dense (so that the 2-norm of the target function is much smaller than its 1-norm), then the Perceptron family of algorithms learn more quickly. This empirical study is consistent with the theoretical results.

To apply the theoretical results to NLP problems, we shall note that for such problems, typically a data point not only has a small infinity norm, but also is sparse. This sparsity implies that the 2-norm of the data is also small, which in turn implies that in theory, Perceptron-like algorithms can perform as well as Winnow-like algorithms. On the other hand, for NLP problems, most data dimension is likely to be not very useful. Although the target function might not be strictly sparse (as argued in [11], non-important features might still have small impacts on classification performance), it is likely to be concentrated on a small set of features, so that the target hyperplane has a relatively small 2-norm and a relatively small 1-norm. In this situation, the Perceptron does not have any theoretical advantage over Winnow. Therefore it is not surprising that for NLP problems, we expect both the Winnow family and the Perceptron family of algorithms work equally well, as demonstrated by our experiments on text-categorization.

In summary, although the argument of many irrelevant features in [5] is somewhat true, the authors neglected the sparsity of the data. The argument in [11] that there are many relevant features in text-categorization seems to be a little misled since unless the features are more or less equally important in the target function, the dominant effect is still near irrelevance for most features. Such an effect can be picked up by both Perceptron and Winnow families of algorithms.

4. TEXT CATEGORIZATION EXAMPLES

In this Section, we compare the classification performance of Perceptron and Winnow families of algorithms on text categorization, which is central to many NLP tasks. We use the standard Reuters-21578 data set which is publicly available from *http://www.research.att.com/~lewis/reuters21578.html*. We use the Mod-Apte split which contains 9603 training data and 3299 test data.

For text categorization especially for Reuters which is multiply categorized, the performance is usually measured by precision and recall rather than classification error. Precision is the percentage of correctly classified data among all data that are classified to be in-class; recall is the percentage of correctly classified data among all data that are truly in-class. Since we can adjust the threshold of a linear classifier after training to trade-off its precision and recall, a popular performance metric is the break-even point (BEP) where the threshold is chosen so that precision equals recall [22]. Another widely used single number metric is the F_1 metric defined as the harmonic mean of the precision and the recall [22]. Typically when the precision and recall are more or less balanced, BEP and the F_1 metric are rather close. In this case, either metric is a very indicative performance measurement. In our experiments, we shall use the BEP measurement since it appears to be more frequently adopted in text categorization. In addition, we will also plot the precision-versus-recall curve separately for a few methods we study here. Since the break-even point is a met-

[1] For example, the data is binary as in [15] and the relevant experiments in [14].

[2] As in [15, 14], this assumption is typically assured by the sparsity of the "target concept". The prior μ is usually chosen to be uniform, which implies an $\log d$ (d is the dimension of x) worst-case entropy factor in their bounds.

ric for binary-classification, we use the micro-average [22] (which uses the statistics by summing over each individual confusion matrix for each binary-categorization problem) to measure the overall performance.

We use binary word occurrences as the input features, and append a constant feature of 1. As explained before, the constant feature is used to offset the effect of the linear threshold θ which we intentionally set to zero. Another popular vector representation for text is the TFIDF weighted word occurrence features, which has been adopted in [11] for SVMs. Although this could give a slight improvement (which we haven't tested), the SVM result of [6] with binary features suggests that such an improvement, if any, will be small. Another reason we use binary features is that it is infinity-norm bounded, thus suitable for Winnows. By using the same set of features for both the Winnow family and the Perceptron family of algorithms, we can eliminate the effect of different feature representations on the performance of different algorithms. We use word-stemming in feature preparation without stop-word removal. We then use a criterion similar to the information gain (IG) method in [23] to select 10000 features, where we replace the entropy scoring in the IG criterion by the Gini-index scoring. Note that both entropy and Gini-index are useful for finding relevant attributes as described in [2]. We use the Gini-index instead of entropy mainly because we do not use stop word removal. Our experience seems to suggest that the Gini-index is more capable to pick up non stop-words, although the difference is relatively small.

We shall only report the balanced versions of the Winnows. For consistency and comparison purposes, we fix the learning rates as 0.01 and use 200 iterations over the training data for all algorithms. The initial values for the Winnows are $\mu_j = 0.01$. We use UWin and NWin to denote the basic unnormalized and normalized Winnows respectively. LM-UWin and LM-NWin are used to denote the respective large margin versions. The regularization parameter C for each algorithm is determined by cross-validation, and is set to a fixed value for all categories. Our results are comparable with a sigmoid enhanced SVM result in [6] (and slightly better than those of more complicated kernel SVMs in [11]), where the micro-averaged break-evens was 92.0 over top 10 categories, and 87.0 over all 118 categories.

It is worth mentioning that in Table 1, UWin and NWin have the same results, which is not a programming error. In this situation, the training data are linearly separable for all categories (since the dimension is larger than the number of data points), and we find that both Winnow methods compute linear separators relatively quickly. Since the learning rate is rather small, therefore the norm of the weight does not change significantly. In particular, both Winnows have the same short sequence of mistakes, therefore they end up with equivalent weights. Note that this phenomenon is not generally true for large margin Winnows, due to the more subtle update rules. Even though normalized Winnows and unnormalized Winnows are comparable, we find that the large margin normalized Winnow seems to be more sensitive to the prior choice μ. Therefore in practice, an unnormalized Winnow could be more robust.

In Figure 1, we have also plotted the precision-recall curves (over the top 10 categories) for SVM, large margin normalized Winnow and Perceptron for comparison purposes. SVM and large margin normalized Winnow are almost indistinguishable.

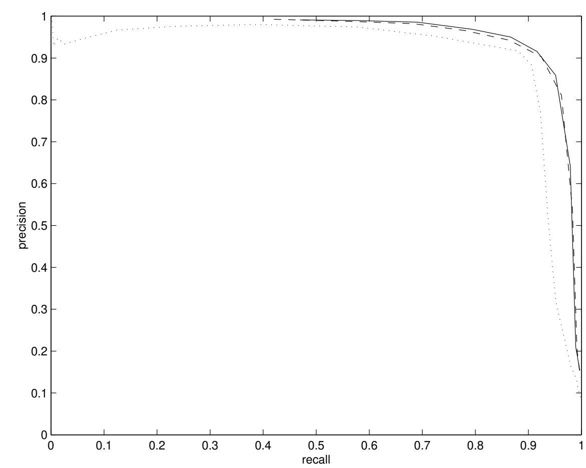

Figure 1: Micro-Avg precision-recall curve for 10 Largest Categories (10000 features). SVM = 'solid'; LM-NWin = 'dashed'; Perceptron= 'dotted'.

In Table 2, we compare the performance of the algorithms without any feature selection (but keep the same setup for other parameters). In this case, stop-words will appear in the features as relatively dense irrelevant attributes. According to our theoretical analysis, these relatively dense irrelevant attributes will give the Winnow family of algorithms a slight advantage over the Perceptron family of algorithms. This phenomenon can be observed in Table 2. Interestingly, most degradation of Perceptron algorithms tends to come from those relatively small categories. Also unregularized algorithms are much more sensitive to feature selection than their regularized versions. Since stop-words can always be removed by using appropriate feature selection techniques, it is therefore unclear whether the slight advantage of Winnow algorithms without feature selection has much practical significance.

If we do not use any feature selection or stop-word removal, then the small advantage of regularized Winnows may also be observed on other data. For example, with the seven category CMU Web-Kb data available at *http://www.cs.cmu.edu /afs/cs.cmu.edu/project/theo-20/www/data/*, if we perform a random 3-fold cross-validation experiment, then without stop-word removal, the regularized Winnows achieve accuracies of about 90%, while an SVM achieves an accuracy of about 88%. Even though it is not clear how statistically significant this difference is, we have observed that with stop-word removal, the SVM performance can be enhanced. For this example, we treat the problem directly as a seven-category classification problem. This is achieved by training linear classifier weights as seven separate binary classification problems, one for each category. The predicted label of a data-point x is the label l with the largest value of $w_l^T x$ where w_l is the linear weight associated with class

category	Perceptron	SVM	UWin	LM-UWin	NWin	LM-NWin
earn	98.3	98.3	98.6	98.4	98.6	98.6
acq	94.0	95.4	93.8	95.0	93.8	95.2
money-fx	71.7	73.7	71.3	74.3	71.3	75.4
grain	87.9	90.6	88.6	92.6	88.6	92.6
crude	83.4	83.6	80.7	83.6	80.7	83.6
trade	70.1	73.5	72.6	71.8	72.6	73.5
interest	68.4	77.1	69.5	75.6	69.5	77.1
ship	75.6	85.4	76.4	84.3	76.4	84.3
wheat	82.3	85.9	85.9	87.3	85.9	87.3
corn	81.1	85.7	82.1	83.9	82.1	85.7
Micro-Avg Top 10	89.8	91.5	90.0	91.5	90.0	91.8
Micro-Avg All 118	83.4	86.7	83.4	86.8	83.4	87.0

Table 1: Break-even Performance for 10 Largest Categories and Micro-Averages (10000 features)

l. To see that this simple approach to multi-class problems is quite reasonable, we shall mention as a comparison that a Naive Bayes classifier (which directly handles multi-class problems) gives an accuracy of about 66% on the same data.

5. CONCLUDING REMARKS

In this paper, we have compared Perceptron family and Winnow family of algorithms for text categorization. From our analysis and experiments, we show that the idea of regularization (or related, margin) is very helpful for text categorization applications. However, due to the sparse structures of the data and the target function, the specific form of regularization (2-norm or entropy) is less important, especially when an appropriate feature selection method is employed. Since text categorization is the basis for many NLP tasks, our study implies that both families of algorithms are equally suitable to these problems.

6. REFERENCES

[1] J. Anlauf and M. Biehl. The AdaTron: an adaptive perceptron algorithm. *Europhys. Lett.*, 10(7):687–692, 1989.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wadsworth Advanced Books and Software, Belmont, CA, 1984.

[3] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[5] I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In *Proceedings of the Second Conference on Empirical Methods in NLP*, 1997.

[6] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 1998 ACM 7th international conference on Information and knowledge management*, pages 148–155, 1998.

[7] A. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. In *Proc. 10th Annu. Conf. on Comput. Learning Theory*, pages 171–183, 1997.

[8] A. Grove and D. Roth. Linear concepts and hidden variables. *Machine Learning*, 2000. To Appear; early version appeared in NIPS-10.

[9] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, to appear.

[10] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 470–476. MIT Press, 2000.

[11] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learing, ECML-98*, pages 137–142, 1998.

[12] R. Khardon, D. Roth, and L. Valiant. Relational learning for NLP using linear threshold elements. In *Proceedings IJCAI-99*, 1999.

[13] W. Kinzel. Statistical mechanics of the perceptron with maximal stability. In *Lecture Notes in Physics*, volume 368, pages 175–188. Springer-Verlag, 1990.

[14] J. Kivinen and M. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Journal of Information and Computation*, 132:1–64, 1997.

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

[16] O. Mangasarian and D. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, 1999.

category	Perceptron	SVM	UWin	LM-UWin	NWin	LM-NWin
earn	98.3	98.3	98.4	98.3	98.4	98.4
acq	93.4	95.3	94.0	95.1	94.0	95.3
money-fx	72.6	73.7	69.3	74.9	69.3	74.3
grain	84.5	87.9	85.4	91.9	85.4	91.9
crude	84.7	85.2	82.5	85.7	82.5	85.7
trade	73.5	76.1	72.6	75.2	72.6	76.1
interest	66.9	71.8	64.4	72.5	64.4	72.5
ship	77.0	82.0	69.4	87.6	69.4	87.6
wheat	77.5	85.9	83.7	85.9	83.7	85.9
corn	74.3	87.5	76.8	89.3	76.8	89.3
Micro-Avg Top 10	89.5	91.2	89.2	91.8	89.2	91.9
Micro-Avg All 118	78.5	85.1	79.2	87.0	79.2	87.1

Table 2: Break-even Performance for 10 Largest Categories and Micro-Averages (no feature selection)

[17] M. Opper. Learning times of neural networks: Exact solution for a perceptron algorithm. *Phys. Rev. A*, 38(7):3824–3826, 1988.

[18] B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods : Support Vector Learning*. The MIT press, 1999.

[19] J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. Inf. Theory*, 44(5):1926–1940, 1998.

[20] V. Vapnik. *Statistical learning theory*. John Wiley & Sons, New York, 1998.

[21] R. C. Williamson, A. J. Smola, and B. Schölkopf. Entropy numbers of linear function classes. In *COLT'00*, pages 309–319, 2000.

[22] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, 1:69–90, 1999.

[23] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[24] T. Zhang. Analysis of regularized linear functions for classification problems. Technical Report RC-21572, IBM, 1999. An early abstract in NIPS'99, pp. 370–376.