

# Using Hypothesis Margin to Boost Centroid Text Classifier

Songbo Tan<sup>1</sup> and Xueqi Cheng<sup>1</sup>

<sup>1</sup> Intelligent Software Department, ICT, P.O. Box 2704, Beijing, 100080, CHINA  
tansongbo@software.ict.ac.cn, cxq@ict.ac.cn

## Abstract

Centroid Classifier is a simple and yet efficient method for text categorization. However it often suffers from the inductive bias or model misfit incurred by its assumption. In order to address this issue, training-set errors as well as training-set margins are regarded as training criterions. Based on these two criterions, an overall (or global) objective function over all training examples is constructed, and optimized to produce a refined Centroid classification model. The empirical assessment conducted on four benchmark collections evidence that proposed method performs comparably to state-of-the-art SVM classifier in classifying performance, as well as beats it in running time.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval-search process; I.2 [Artificial Intelligence]: Learning

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Text Classification, Data Mining, Information Retrieval, Machine Learning

## 1 INTRODUCTION

Due to the rapid explosion of texts in digital form, text categorization has become an important area of research owing to the need to automatically organize and index large text collections in various ways. A large number of machine learning techniques have been applied to text classification problem, including Centroid Classifier [1-2], K-Nearest Neighbor (KNN) [3-5], Rocchio [6-7], Naive Bayes [8-10], Winnow [11-12], Perceptron [11-12], Voting [13-15] and Support Vector Machines (SVM) [16-17].

Among all these methods, Centroid Classifier is a simple and yet efficient method for text categorization. However it often suffers from the inductive bias [19] or model misfit [18] incurred by its assumption. For example, Centroid Classifier makes a simple assumption that a given document should be assigned a particular class if the similarity of this document to the centroid of the class is the largest. However, this supposition is often violated (misfit) when there exists a document from class *A* sharing more similarity with the centroid of class *B* than that of class *A*.

In order to address this issue, numerous researchers have

devoted their energy to refining the classifier model by proposing efficient strategies. One of the popular strategies is Voting [13-15], which takes one classifier and training set as input and trains the classifier multiple times on different versions of the training set. Wu et al. [18] presented another novel approach to deal with the problem of model misfits. Their technique needs to train multiple sub-classifiers using misclassified training examples of each predicted class with the same learning method.

However, both above refinement approaches employ only one criterion, i.e., training-set error, as its objective function. From the point of view of machine learning, training-set error based objective function cannot guarantee the generalization capability of base classifiers and may lead to over-train over training data.

With the aim of solving this problem, this paper introduces the idea of measuring the margin that base classifiers induced. As a result, in order to combine these two criterions, an overall (or global) objective function over all training examples is constructed, and optimized to produce a refined Centroid classification model. For the sake of convenience, we refer to this refinement strategy as “Hypothesis-Margin Based Global Refinement (HMGR)”.

Extensive experiments conducted on four benchmark document corpora show that the proposed technique is able to improve classification performance of Centroid Classifier dramatically. The resulting classifier not only approaches state-of-the-art SVM in classifying performance, but also beats it in running time.

The rest of this paper is constructed as follows: Next section describes Centroid Classifier. Hypothesis-margin under the framework of Centroid Classifier is defined in section 3. Proposed techniques HMGR1 and HMGR2 are introduced in the section 4. Experimental results are given in Section 5. Finally section 6 concludes this paper.

## 2 CENTROID CLASSIFIER

In this work, the documents are represented using vector space model. In this model, each document *d* is considered to be a vector in the term-space. For term weight we employ Normalized TFIDF method [28] as formula (1). This function embodies the intuitions that the more often a term occurs in a document, the more it is representative of its content, and the more documents a term occurs in, the less discriminating it is.

$$w(t, d) = \frac{tf(t, d) \times \log(D / n_t)}{\sqrt{\sum_{t \in d} [tf(t, d) \times \log(D / n_t)]^2}} \quad (1)$$

where *D* is the total number of training documents, *n<sub>t</sub>* is the number of documents containing the word *t*, and *tf(t,d)* indicates the occurrences of word *t* in document *d*. Obviously, formula (1) calculates TFIDF [27] for each document and then normalize it by 2-norm.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07, March 11-15, 2007, Seoul, Korea.

Copyright 2007 ACM 1-59593-480-4/07/0003...\$5.00.

After giving the representation of documents, we can compute centroid by summing document vectors in class  $C_i$ :

$$\vec{C}_i = \sum_{d \in C_i} \vec{d} \quad (2)$$

Then we count the similarity of one document  $d$  to each centroid by cosine measure as following,

$$Sim(d, C_i) = \frac{\vec{d} \cdot \vec{C}_i}{\|\vec{d}\|_2 \cdot \|\vec{C}_i\|_2} = \frac{\vec{d} \cdot \vec{C}_i}{\|\vec{C}_i\|_2^2} \quad (3)$$

where  $\|z\|_2$  denotes the 2-norm of  $z$ .

Lastly, based on these similarities, we assign  $d$  the class label corresponding to the most similar centroid:

$$C = \arg \max_{C_i} (sim(d, C_i)) = \arg \max_{C_i} \left( \frac{\vec{d} \cdot \vec{C}_i}{\|\vec{C}_i\|_2^2} \right) \quad (4)$$

### 3 HYPOTHESIS MARGIN

Margins play a crucial role in the modern machine learning research. They measure the classifier confidence when making its decision. There are two types of margins [21]. The more common type, i.e., sample-margin, measures how far the positive and negative training examples are separated by the decision surface. Based on this sample-margin conception, Vapnik [22] defined the objective function in SVM as both minimization of training-set error and maximization of margin width. Just so, SVM has been proved to be a top-performer in text categorization [3].

An alternative definition, hypothesis-margin, requires the existence of a distance measure on the hypothesis class. The margin of a hypothesis with respect to instance is the distance between the hypothesis and the closest hypothesis that assigns alternative label to given instance. For example, the 1-NN classifier computes the hypothesis-margin [21][23] of instance  $x_p$  with respect to a set of points  $P$  by following definition,

$$HM_p(x_p) = \frac{1}{2} (\|x_p - x_R\| - \|x_p - x_M\|) \quad (5)$$

where  $x_R$  and  $x_M$  denote the nearest point to  $x_p$  in  $P$  with the same and different label, respectively.

Similar to above definition of hypothesis-margin, we introduce an evaluation function which assigns a score to a Centroid Classifier according to the margin it induces. First we formulate the margin of one instance  $x_p$  with respect to a Centroid Classifier  $C$  as following,

$$HM(x_p, C_R, C_M) = (Sim(x_p, C_R) - Sim(x_p, C_M)) \quad (6)$$

where  $C_R$  and  $C_M$  denote the most similar Centroid to  $x_p$  with the same and different label, respectively.

Now we turn to define the evaluation function. The building blocks of this function are the margins of all training examples. Given a training set  $S$ , the evaluation function can be deduced as following formula,

$$GHM(S, C) = \sum_{x_p \in S} (Sim(x_p, C_R) - Sim(x_p, C_M)) \quad (7)$$

According to Centroid Classifier decision rule (formula (4)), if  $HM(x_p, C_R, C_M) < 0$ , that is to say,  $Sim(x_p, C_R) < Sim(x_p, C_M)$ , the instance  $x_p$  will be misclassified. In this case, the instance  $x_p$  serves as one training error. In order to balance the training errors and training margins in

the objective function, we introduce a constant parameter "Weight". Consequently we modified the formula (7) as following,

$$GHM(S, C) = \sum_{x_p \in S} Weight \times |Sim(x_p, C_R) - Sim(x_p, C_M)|_- + \sum_{x_p \in S} |Sim(x_p, C_R) - Sim(x_p, C_M)|_+ \quad (8)$$

where  $|z|_+$  denotes the hinge function which equals to the argument  $z$  if  $z > 0$  and zero otherwise,  $|z|_+ = \max\{z, 0\}$ , and contrariwise  $|z|_- = \min\{z, 0\}$ .

As we all know,  $Sim(x, y)$  ranges from 0 to 1 according to formula (3) and the margin  $HM(x_p, C_R, C_M)$  ranges from  $-1$  to  $1$ . If example margin nears 0, we say the margin is quite small and it needs enlarge; On the other hand, if example margin approaches 1, we say the margin is very large and it may not need increase. Accordingly, in order to concentrate our attention on small-margin examples, we introduce a small positive margin threshold, *MinMargin* (we substitute it with a symbol  $\theta (> 0)$ ), and update formula (8) as (9),

$$GHM(S, C) = \sum_{x_p \in S} Weight \times |Sim(x_p, C_R) - Sim(x_p, C_M)|_- + \sum_{x_p \in S} |Sim(x_p, C_R) - Sim(x_p, C_M)|_{+, \theta} \quad (9)$$

where

$$|z|_{+, \theta} = \begin{cases} \max\{z, 0\} & \text{if } z \leq \theta \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

## 4 PROPOSED METHOD

In this section, we present two algorithms that attempt to minimize the training-set error and maximize the training-set margin based evaluation function.

### 4.1 The Derivation of Gradient

According to formula (3), formula (6) can be written as following,

$$HM(x_p, C_R, C_M) = \left( \frac{x_p \cdot C_R}{\|C_R\|_2} - \frac{x_p \cdot C_M}{\|C_M\|_2} \right) \quad (11)$$

Given a training corpus with  $K$  classes and  $W$  words. We denote the  $K$  centroids as  $\{\vec{C}_1, \vec{C}_2, \dots, \vec{C}_K\}$  where  $C_k = (c_{k1}, c_{k2}, \dots, c_{kW})$ . Since the evaluation function  $GHM(S, C)$  is smooth almost everywhere, we can employ gradient ascent in order to maximize it.

For each  $c_{ki}$ , we can derive the gradient of  $HM(x_p, C_R, C_M)$  as following,

$$\frac{\partial(HM(x_p, C_R, C_M))}{\partial c_{ki}} = \begin{cases} \frac{x_{pi}}{\|C_k\|_2} - \frac{c_{ki}}{\|C_k\|_2^3} (x_p \cdot C_k) & \text{if } R = k \\ -\frac{x_{pi}}{\|C_k\|_2} + \frac{c_{ki}}{\|C_k\|_2^3} (x_p \cdot C_k) & \text{if } M = k \end{cases} \quad (12)$$

The gradient of  $GHM(S, C)$  when evaluated on a training set  $S$  is:

$$\frac{\partial(GHM(S, C))}{\partial(c_{ki})} = \sum_{\substack{R=k \\ x_p \in S}} \frac{\partial(HM(x_p, C_R, C_M))}{\partial c_{ki}}$$

$$+ \sum_{\substack{M=k \\ x_p \in S}} \frac{\partial(HM(x_p, C_R, C_M))}{\partial c_{ki}} \quad (13)$$

leads to

$$\begin{aligned} \frac{\partial(GHM(S, C))}{\partial(c_{ki})} = & \sum_{\substack{R=k \\ x_p \in S}} \left( \frac{x_{pi}}{\|C_k\|_2} - \frac{c_{ki}}{\|C_k\|_2^3} (\bar{x}_p \cdot \bar{C}_k) \right) \\ & + \sum_{\substack{M=k \\ x_p \in S}} \left( -\frac{x_{pi}}{\|C_k\|_2} + \frac{c_{ki}}{\|C_k\|_2^3} (\bar{x}_p \cdot \bar{C}_k) \right) \end{aligned} \quad (14)$$

then we obtain formula (15),

$$\begin{aligned} \frac{\partial(GHM(S, C))}{\partial(c_{ki})} = & \frac{1}{\|C_k\|_2} \left( \sum_{\substack{R=k \\ x_p \in S}} (x_{pi}) - \sum_{\substack{M=k \\ x_p \in S}} (x_{pi}) \right) \\ & + \frac{c_{ki}}{\|C_k\|_2^3} \left( \left( \sum_{\substack{M=k \\ x_p \in S}} (\bar{x}_p) - \sum_{\substack{R=k \\ x_p \in S}} (\bar{x}_p) \right) \cdot \bar{C}_k \right) \end{aligned} \quad (15)$$

finally, gradient updated formula can be written as

$$c_{k,i}^{o+1} = \left| c_{k,i}^o + LearnRate \times \frac{\partial(GHM(S, C))}{\partial(c_{ki})} \right|_+ \quad (16)$$

where  $o$  denotes current iteration-step, i.e., the  $o^{\text{th}}$  iteration-step, and *LearnRate* controls the amount of update applied to the base classifier.

## 4.2 The HMGR1 Algorithm

**Initialization:** to start, we need to load the training data and parameters. Then for each category  $C_k$  we calculate one summed centroid  $C_k^o$ . Note that  $o$  denotes current iteration-step, i.e., the  $o^{\text{th}}$  iteration-step.

**Updating:** in one iteration, first we calculate the similarities of each example  $x_p$  to  $K$  centroids and pick  $C_M$  for the example. Then for each centroid  $C_k$ , and for each element  $c_{ki}^o$  of  $C_k^o$ , we update it by formula (16). Note that we employ  $|\cdot|_+$  to substitute the negative with zero since we found that nonnegative centroid performs better than real centroid in our experience.

**Time Requirements:** Assume that there are  $D$  training documents,  $T$  test documents,  $W$  words,  $K$  classes and *Miter* iteration. The time complexity of calculating one summed centroid for each class is  $O(DW+KW)$ . Since  $D>K$ , the time complexity is  $O(DW)$ . In each iteration of updating phase, for each example  $x_p$ , we need to calculate its similarities to  $K$  centroids and this calculation takes  $O(KW)$ . The selection of  $C_M$  for each example can be done in  $O(K-1)$ . Then for all examples the running time is  $O(D(KW+K-1))$ , i.e.,  $O(DKW)$ . The computation of gradient for all centroids takes  $O(2DW)$ . The updating of all centroids requires  $O(KW)$ . Consequently, the total running time is  $O(Miter(DKW+2DW+KW)+DW)$ , i.e.,  $O(MiterDKW)$ . As a result, the training time of HMGR1 scales linearly with the training documents ( $D$ ). Since the final classifier obtained by HMGR1 still consists of  $K$  centroids, the prediction time required by refined classifier is the same as Centroid Classifier, i.e.,  $O(TKW)$ . Accordingly HMGR1 is still a linear classifier.

## 4.3 The HMGR2 Algorithm

**Initialization:** As depicted in Figure 2, the only difference from HMGR1 in this phase is that HMGR2 need to compute the similarities of each example to  $K$  centroids.

**Updating:** in one iteration, unlike HMGR1, HMGR2 update one centroid  $C_k^o$  and then update the similarities of all training examples with respect to  $C_k$  and finally pick the  $C_M$  for each example, and so on.

**Time Requirements:** The time complexity of training a Centroid Classifier is  $O(DW)$  and the computation of the similarities of each example to  $K$  centroids takes  $O(DKW)$ . The running time of updating phase is  $O(Miter(K(2DW+DW+D(K-1))))$ , i.e.,  $O(MiterDKW)$ . Consequently, the total training time is  $O(MiterDKW+DW+DKW)$ , i.e.,  $O(MiterDKW)$ . As a result, the training time of HMGR2 also scales linearly with the training documents ( $D$ ).

---

```

Load training data and parameters;
Calculate  $C_k^o$  for each class;
For iter=1 to MaxIteration Do
  For p=1 to D Do
    Calculate the similarities of  $x_p$  to each class
    Calculate  $C_R$  for  $x_p$ 
  End For
  For k=1 to K Do
    For i=1 to W Do
       $c_{k,i}^{o+1} = \left| c_{k,i}^o + LearnRate \times \frac{\partial(GHM(S, C))}{\partial(c_{ki})} \right|_+$ 
    End For
  End For
End For

```

---

Figure 1: The Outline of HMGR 1 Algorithm

---

```

Load training data and parameters;
Calculate  $C_k^o$  for each class;
For p=1 to D Do
  Calculate the similarities of  $x_p$  to each class
  Calculate  $C_R$  for  $x_p$ 
End For
For iter=1 to MaxIteration Do
  For k=1 to K Do
    For i=1 to W Do
       $c_{k,i}^{o+1} = \left| c_{k,i}^o + LearnRate \times \frac{\partial(GHM(S, C))}{\partial(c_{ki})} \right|_+$ 
    End For
  For p=1 to D Do
    Update the similarities of  $x_p$  to the class  $C_k$ 
    Calculate  $C_R$  for  $x_p$ 
  End For
End For
End For

```

---

Figure 2: The Outline of HMGR2 Algorithm

## 4.4 Comparison to Related Approaches

As everyone knows, SVM is a classical margin-based classifier. The core of SVM is to find a decision surface that

“best” separates the data points into two classes. Specifically, the “best” decision surface in a linearly separable space is a hyperplane that maximizes the “margin”, that is the distance between two parallel hyperplanes that separate the two classes of data points in the training set.

In contrast to standard SVM classifier, HMGR has the following two characteristics. First, HMGR starts from centroid classifier while SVM starts from a random-selected point; Second, HMGR employs hypothesis margin (or approximate margin) while SVM use sample margin (or accurate margin). The two merits account for the fact that HMGR runs much faster than SVM.

Voting is a famous strategy for correction of inductive bias. It works by taking a classifier and training set as input and training the classifier multiple times on different versions of the training set. The generated classifiers are then combined to create a final classifier that is used to classify the test set.

Compared to Voting, HMGR has two particularities. First, unlike Voting, our technique does not need to retrain the classifier multiple times on the different versions of the entire training set. Consequently our approach consumes much less training time than Voting method. Second, HMGR produces only one refined classifier. Hence the prediction is much faster than Voting method.

Wu et al. [18] presented another novel approach to handle the problem of model misfits. Once a prediction error appears on a training set, their technique retrains a sub-classifier using these misclassified training examples of each predicted class with the same learning method. In this way, it forces the classifier to learn from refined regions in the training data, making the model stronger and fit the training data better. As a result, their method leads to a classifier-model tree consisting of a large number of nodes (each node stands for a classifier).

Unlike Wu’s technique, HMGR does not need to train multiple classifiers on multiple subsets of the entire training set, and instead it produces only one refined classifier. Meanwhile, HMGR does not need to split the training set; consequently the unbalance of the data set exerts less adverse influence on HMGR than Wu’s technique. Furthermore, Wu’s technique utilizes only training-set errors while HMGR employs training-set errors as well as training-set margins.

## 5 EMPIRICAL ASSESSMENT

### 5.1 The Datasets

In our experiment, we use four corpora: Reuter-21578<sup>1</sup>, 20NewsGroup<sup>2</sup>, Industry Sector<sup>3</sup> and OHSUMED<sup>4</sup>.

**Reuter-21578** The Reuters-21578 text categorization test collection contains documents collected from the Reuters newswire in 1987. It is a standard text categorization benchmark and contains 135 categories. We used its subset: one consisting of 92 categories and in total 10,346 documents.

**20NewsGroup** The 20Newsgroup (20NG) dataset contains approximately 20,000 articles evenly divided

among 20 Usenet newsgroups. We use a subset consisting of total categories and 19,446 documents.

**Sector-48** The Industry Section dataset is based on the data made available by Market Guide, Inc. (www.marketguide.com). The set consists of company homepages that are categorized in a hierarchy of industry sectors, but we disregard the hierarchy. There were 9,637 documents in the dataset, which were divided into 105 classes. We use a subset called as Sector-48 consisting of 48 categories and in all 4,581 documents.

**OHSUMED** The OHSUMED [24] dataset is a bibliographical document collection: developed by William Hersh and colleagues at the Oregon Health Science University, Which is a subset of MEDLINE database. We use a subset (called ohscal<sup>5</sup> in [2]) from OHSUMED dataset that contains 11,162 documents and in total 10 categories: Antibodies, Carcinoma, DNA, In-Vitro, Molecular-Sequence-Data, Pregnancy, Prognosis, Receptors, Risk-Factors and Tomography.

### 5.2 Experimental Design

We evenly split the each dataset into three parts. Then we use two parts for training and the remaining third for test. We perform the train-test procedure three times and use the average of the three performances as final result. This is so called three-fold cross validation.

To evaluate a text classification system, we use the Micro- and Macro-averages of F1 scores [26]. The MicroF1 emphasizes common categories when calculating the performance of the system; while MacroF1 emphasizes rare categories. Using these averages, we can observe the effect of different kinds of data on a text classification system.

In order to remove redundant features and save running time, we employ Information Gain as feature selection method because it consistently performs well in most cases [25].

Algorithms are coded in C++ and running on a Pentium-4 machine with 2.0GHz CPUs.

For experiments involving SVM we employed SVMtorch, which uses one-versus-the-rest decomposition and can directly deal with multi-class classification problems. (<http://www.idiap.ch/~bengio/projects/SVMtorch.html>).

In our experiments, we only run Balanced Winnow for it consistently yields better performance than Positive Winnow [11]. The Balanced Winnow keeps two weights for each feature  $l$  in category  $C_i$ ,  $w_{il}^+$  and  $w_{il}^-$ . The weight values are initialized as  $w_{il}^+=2.0$  and  $w_{il}^-=1.0$  and the threshold was set to 1.0. The promotion parameter  $\alpha$  and the demotion  $\beta$  (learning rates) were fixed as 1.2 and 0.8 respectively. We train Balanced Winnow for 40 rounds over the training data. For the sake of brevity, we substitute Balanced Winnow with Winnow.

For KNN, we set the neighbor number  $k$  to 13. It is worth noticing that except for Winnow and SVM we do not introduce any thresholds investigated by Yang [20] because the adjusting of thresholds may incur significant computational costs.

1 <http://www.research.att.com/~lewis/reuters21578.html>.

2 <http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/wkbb>.

3 <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/>.

4 <http://medir.ohsu.edu/pub/OHSUMED/>.

5 <http://www.cs.umn.edu/~han/data/tmdata.tar.gz>.

### 5.3 Comparison and Analysis

Very surprisingly, HMGR1 and HMGR2 produce nearly the same performance on four datasets. As a result, we use HMGR to stand for HMGR1 and HMGR2 for the sake of simplicity.

Table 1: The Best MicroF1 of different methods

	HMGR1	HMGR2	Centroid	KNN	Winnow	SVM Torch
OHSUMED	0.8014	0.8025	0.7676	0.6494	0.7193	0.7565
Reuter	0.8381	0.8371	0.7820	0.8323	0.8263	0.8706
Sector-48	0.9125	0.9138	0.8055	0.8193	0.8003	0.9026
20NG	0.8724	0.8727	0.8429	0.8503	0.8105	0.8891

Table 2: The Best MacroF1 of different methods

	HMGR1	HMGR2	Centroid	KNN	Winnow	SVM Torch
OHSUMED	0.7911	0.7923	0.7600	0.6363	0.7110	0.7433
Reuter	0.5916	0.5894	0.5617	0.5296	0.4891	0.6013
Sector-48	0.9150	0.9163	0.8152	0.8238	0.8389	0.9049
20NG	0.8680	0.8684	0.8389	0.8487	0.8161	0.8876

Table 1 and Table 2 show the best-performance comparison in MicroF1 and MacroF1. Note that for HMGR, *MaxIteration*, *Weight*, *LearnRate*, and *MinMargin* are set to 80, 5, 1, and 1 respectively. HMGR outperforms all the other four methods on OHSUMED and Sector-48. SVM performs the best on Reuter-21578 and 20NewsGroup. HMGR improves the performance of Centroid Classifier dramatically, and the improvement is especially significant on Sector-48.

On Sector-48, the MicroF1 of HMGR1 is 91.25%, which is approximately 9% higher than that of KNN, 11% higher than that of Centroid and Winnow, and 1% higher than SVM. On OHSUMED, the MicroF1 of HMGR1 beats KNN by approximately 15%, Centroid by 3% and Winnow by approximately 8%, and SVM by 4%. In total HMGR yields excellent performance approaching SVM. Consequently we can say that HMGR is an efficient and competitive algorithm in text classification.

Table 3 reports the training time of six methods on four text collections. Note that the running time does not include the seconds for loading data from hard disk. Feature number is set to 10,000. For HMGR, *MaxIteration*, *Weight*, *LearnRate*, and *MinMargin* are set to 80, 5, 1, and 1 respectively. The training CPU time required by SVM is about 30 times larger than that of HMGR on OHSUMED, and about 20 times larger on 20NewsGroup. Consequently, the time saving of HMGR is very obvious. In summary, these experiments have shown that HMGR offers alternate choice for text categorization.

Table 3: Training Time in seconds

	HMGR1	HMGR2	Centroid	KNN	Winnow	SVM Torch
OHSUMED	6.037	7.709	0.349	0	3.04	229.219
Reuter	73.552	100.214	0.567	0	10.422	131.187
Sector-48	37.453	44.188	0.724	0	6.630	97.677
20NG	25.031	27.063	0.703	0	8.442	564.291

Figure 3, 4, 5 and 6 show training error, margin and performance curves of HMGR vs. *MaxIteration* on Sector-48. Note that feature number takes 10,000. And for HMGR, *Weight*, *LearnRate*, and *MinMargin* are set to 5, 1, and 1 respectively.

The first observation is that the refinement operation based on Hypothesis-Margin can decrease training error, enlarge margin and boost prediction performance. *MaxIteration* equivalent to 0 means that no updating operation is used at all, i.e., Centroid Classifier. From figure 6 we can observe that a wide margin improvement is achieved by running only 20 round of refinement operation over training set.

The second phenomenon is that when iteration is larger than 20, the training error keeps unchanged while the margin and prediction performance continue to increase consistently. This phenomenon validates that large margin can improve the prediction performance even if the training error keeps unchanged.

## 6 SUMMARY AND FURTHER RESEARCH

In order to remedy the deficiency of training errors for classifiers refinement, this paper introduces the idea of measuring the quality of classifiers by the margin it induced. Based on this idea, we propose an effective and yet efficient refinement strategy, i.e., Hypothesis-Margin Based Global Refinement (HMGR), to enhance the performance of Centroid Classifier. Our main research contributions are:

We present the definition of Hypothesis-Margin under the framework of Centroid Classifier and employ this definition to formulate an overall objective function. Then we refine the Centroid Classifier by making use of gradient ascent to maximize objective function.

According to the gradient formula, we design two algorithms, i.e., HMGR1 and HMGR2, to refine the base classifier. The two methods both scale linearly with the corpus examples. The main difference is the way they update centroids of base classifier.

Extensive experiments are conducted on four benchmark evaluation collections and the results show that HMGR can make a significant difference on the performance of Centroid Classifier and deliver top performance comparable to state-of-the-art SVM.

The results reported here are not necessarily the best that can be achieved. Our future effort is to seek new techniques to enhance the performance of HMGR and to apply HMGR to other classifiers.

## 7 REFERENCES

- [1] E. Han and G. Karypis. Centroid-Based Document Classification Analysis & Experimental Result. PKDD 2000.
- [2] Shrikanth Shankar and George Karypis. Weight adjustment schemes for a centroid-based classifier. TextMining Workshop, KDD, 2000.
- [3] Y. Yang, X. Lin. A re-examination of text categorization methods. SIGIR. 1999, 42-49.
- [4] Y. Yang. An evaluation of statistical approaches to text categorization. Information Retrieval, 1999, 1(1): 76-88.
- [5] Masand, B., Linoff, G. and Waltz., D. Classifying news stories using memory based reasoning. SIGIR. 1992, 59-64.

- [6] Thorsten Joachims. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. ICML. 1997, 143-151.
- [7] Robert Schapire, Yoram Singer and Amit Singhal. Boosting and Rocchio Applied to Text Filtering (1998). SIGIR. 1998, 215-223.
- [8] D. D. Lewis. Naive (Bayes) at forty: the independence assumption in information retrieval. ECML. 1998, 4-15.
- [9] T. Kalt and W. B. Croft. A new probabilistic model of text classification and retrieval. Tech. Rep. TR98-18, University of Massachusetts Center for Intelligent Information Retrieval, 1996.
- [10] Andrew McCallum, Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. AAAI/ICML-98 Workshop on Learning for Text Categorization[C]. Menlo Park, CA: AAAI Press. 1998, 41-48.
- [11] P.P.T.M. van Mun. Text Classification in Information Retrieval using Winnow.
- [12] T., Zhang. Regularized Winnow Methods. Advances in Neural Information Processing Systems. 2001, 703-709.
- [13] Schapire, R. E. and Singer, Y. Boostexter: A boosting-based system for text categorization. Machine Learning, 2000,39, 135-168.
- [14] R.E. Schapire, Y. Freund, P. Bartlett and W.S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Method. The Annals of Statistics. 1998, 26(5): 1651-1686.
- [15] Kjersti Aas, Line Eikvil. Text Categorisation: A Survey. Raport NR 941, Norwegian Computing Center, 1999, 15.
- [16] T. Joachims. Text categorization with support vector machines: learning with many relevant features. ECML. 1998, 137-142.
- [17] Godbole, S., Sarawagi, S. and Chakrabarti, S. Scaling multi-class support vector machine using inter-class confusion. SIGKDD. 2002, 513-518.
- [18] Haoran Wu, Tong Heng Phang, Bing Liu and Xiaoli Li. A Refinement Approach to Handling Model Misfit in Text Categorization. SIGKDD. 2002, 207-216.
- [19] Yan Liu, Yiming Yang and Jaime Carbonell. Boosting to Correct Inductive Bias in Text Classification. CIKM. 2002,348-355.
- [20] Y. Yang. A study on thresholding strategies for text categorization. SIGIR 2001, 137-145.
- [21] Cramer, K., Gilad-Bachrach, R., Navot, A., & Tishby, N. (2002). Margin analysis of the l<sub>1</sub>q algorithm. Proc. 17th Conference on Neural Information Processing Systems.
- [22] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20, 273-297.
- [23] Ran Gilad-Bachrach, Amir Navot and Naftali Tishby. Margin Based Feature Selection - Theory and Algorithms. ICML, 2004.
- [24] W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. SIGIR. 1994, 192-201.
- [25] Y. Yang and Jan O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. ICML. 1997, 412-420.
- [26] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. SIGIR. 1996, 298-306.
- [27] SALTON, G. AND BUCKLEY, C. 1988. Term-weighting approaches in automatic text retrieval. Inform. Process. Man. 24, 5, 513-523.
- [28] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1-47, March 2002.

