Efficient Text Classification by Weighted Proximal SVM[*]

Dong Zhuang[1], Benyu Zhang[2], Qiang Yang[3], Jun Yan[4], Zheng Chen[2], Ying Chen[1]
[1] *Computer Science and Engineering, Beijing Institute of Technology, Beijing 100081, China*
{zhuangdong, chenying1}@bit.edu.cn
[2]*Microsoft Research Asia, Beijing 100080, China*
{byzhang, zhengc}@microsoft.com
[3]*Computer Science, Hong Kong University of Science and Technology, Hong Kong*
qyang@cs.ust.hk
[4]*Department of Information Science, School of Mathematical Science, Peking University*
yanjun@math.pku.edu.cn

Abstract

In this paper, we present an algorithm that can classify large-scale text data with high classification quality and fast training speed. Our method is based on a novel extension of the proximal SVM mode [3]. Previous studies on proximal SVM have focused on classification for low dimensional data and did not consider the unbalanced data cases. Such methods will meet difficulties when classifying unbalanced and high dimensional data sets such as text documents. In this work, we extend the original proximal SVM by learning a weight for each training error. We show that the classification algorithm based on this model is capable of handling high dimensional and unbalanced data. In the experiments, we compare our method with the original proximal SVM (as a special case of our algorithm) and the standard SVM (such as SVM light) on the recently published RCV1-v2 dataset. The results show that our proposed method had comparable classification quality with the standard SVM. At the same time, both the time and memory consumption of our method are less than that of the standard SVM.

1. Introduction

Automatic text classification involves first training a classifier by some labeled documents and then using the classifier to predict the labels of unlabeled documents. Many methods have been proposed to solve this problem. SVM (Support Vector Machine), which is based on the statistical learning theory [11], has been shown to be one of the best methods for text classification problems [6] [8]. Much research has been done to make SVM practical to classify large-scale dataset [4] [10]. The purpose of our work is to further advance the SVM classification technique for large-scale text data that are unbalanced. In particular, we show that when the text data are largely unbalanced, that is, when the positive and negative labeled data are in disproportion, the classification quality of standard SVM deteiorates. This problem has been solved using cross-validation based methods. But cross-validation methods are very inefficient due to their tedious parameter adjustment routines. In response, we propose a weighted proximal SVM (WPSVM) model, in which the weights can be adjusted, to solve the unbalanced data problem. Using this weighted proximal SVM method, we can achieve the same accuracy as the traditional SVM while requiring much less computational time.

Our WPSVM model is an extended version of the proximal SVM (PSVM) model. The original proximal SVM was proposed in [3]. According to the experimental results of [3], when classifying low dimensional data, training a proximal SVM is much faster than training a standard SVM and the classification quality of proximal SVM is comparable with the standard SVM. However, the original proximal SVM is not suitable for text classification because of the following two reasons: 1), text data are high dimensional data, but the method proposed in [3] is not suitable for training high dimensional data; 2), data are often unbalanced in text classification, but proximal SVM does not work well in this situation. Moreover, in the experiments we found that the classification quality of proximal SVM deteriorates more quickly than standard SVM when the training data becomes unbalanced.

[*] This work is done at Microsoft Research Asia.

1550-4786/05 $20.00 © 2005 IEEE

In response, we propose a weighted proximal SVM (WPSVM) model in this paper. We show that this method can be successfully applied to classifying high dimensional and unbalanced text data through the introduction of the following two modifications: 1) in WPSVM, we added a weight for each training error and developed a simple method to estimate the weights. We then adjusted the weights automatically solves the unbalanced data problem; 2) Instead of solving the problem by KKT (Karush-Kuhn-Tucker) conditions and Sherman-Morrison-Woodbury formula as shown in [3], we use a iterative algorithm to solve WPSVM, which makes WPSVM suitable for classifying high dimensional data.

Experimental results on RCV1-v2 [7] [8] show that the classification quality of WPSVM are as accurate as traditional SVM and more accurate than proximal SVM when the data are unbalanced. At the same time WPSVM is much more computationally efficient than traditional SVM.

The rest of this paper is organized as follows. In Section 2, we review the text classification problems and the SVM and proximal SVM algorithms. In Section 3, we propose the weighted proximal SVM model and explore how to solve it efficiently. In Section 4, we discuss the implementation issues. Experimental results are given in Section 5. In Section 6, we give the conclusions and future work.

2. Problem Definition and Related Work

2.1. Problem Definition

In our formulation, text documents are represented in the Vector Space Model [1]. In this model, each document is represented by a vector of weighted term frequencies using the TF*IDF [1] indexing schema.

For simplicity we first consider the binary classification problem, where there are only two class labels in the training data: positive (+1) and negative (-1). Note that multi-class classification problem can be solved by combining multiple binary classifiers; this will be done in our future work. Suppose that there are *m* documents and *n* terms in the training data, we use $< x_i , y_i >$ to denote each training data, where $\boldsymbol{x}_i \in R^n, i = 1, 2, ..., m$ are training vectors and $y_i \in \{+1, -1\}, i = 1, 2, ...m$ are their corresponding class labels The binary text classification problem can be formulated as follows,

Given a training dataset $\{< \boldsymbol{x}_i , y_i > | \boldsymbol{x}_i \in R^n, y_i \in \{-1,1\}, i = 1, 2...m\}$*, finding a classifier* $f(\boldsymbol{x}): R^n \rightarrow \{+1, -1\}$ *, such that for any unlabeled data* $\boldsymbol{x}$ *, we can predict the label of* $\boldsymbol{x}$ *by* $f(\boldsymbol{x})$.

We first review the standard SVM and proximal SVM. More details could be found in [2] and [3]. This paper will follow the notations of [2] which may differ somewhat from those used in [3]. The SVM algorithms introduced in this paper all use the linear kernel; it is also possible to use non-linear kernels, but there are no significant advantages of using non-linear kernel for text classification.

2.2. Standard SVM Classifier

The standard SVM algorithm aims to find an optimal hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ and use this hyperplane to separate the positive and negative data. The classifier can be written as:

$$f(\boldsymbol{x}) = \begin{cases} +1, & \text{if } \boldsymbol{x} \cdot \boldsymbol{w} + b \geq 0 \\ -1, & \text{if } \boldsymbol{x} \cdot \boldsymbol{w} + b < 0 \end{cases}$$

The separating hyperplane is determined by two parameters $\boldsymbol{w}$ and b . The objective of the SVM training algorithm is to find $\boldsymbol{w}$ and b from the information in the training data. Standard SVM algorithm finds $\boldsymbol{w}$ and b by solving the following optimization problem.

$$min\ \frac{1}{2} \| \boldsymbol{w} \|^2 + C \sum_i \xi_i \qquad (1)$$

$$\text{s.t. } \forall i,\ y_i (\mathbf{w} \cdot \mathbf{x}_\text{i} + b) + \xi_i \geq 1$$

$$\xi_i \geq 0$$

The first term $\| \boldsymbol{w} \|^2$ controls the margin between the positive and negative data. ξ_i represents the training error of the i^{th} training example. Minimizing the objective function of (1) means minimizing the training errors and maximizing the margin simultaneously. *C* is a parameter that controls the tradeoff between the training errors and the margin.

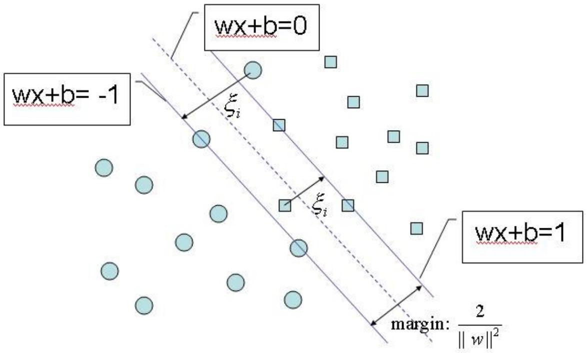

Figure 1. Standard SVM

The intuition of standard SVM is shown in Figure 1. $\boldsymbol{w}\cdot\boldsymbol{x}_i+b=1$ and $\boldsymbol{w}\cdot\boldsymbol{x}_i+b=-1$ are two bounding planes. The distance between the two bounding planes is the margin. The optimization problem (1) can be converted to a standard Quadratic Programming problem. Many efficient methods have been proposed to solve this problem on large scale data [2] [4].

2.3. Proximal SVM Classifier

The proximal SVM also uses a hyperplane $\boldsymbol{w}\cdot\boldsymbol{x}+b=0$ as the separating surface between positive and negative training examples. But the parameter $\boldsymbol{w}$ and b are determined by solving the following problem.

$$\min \frac{1}{2}(\|\boldsymbol{w}\|^2+b^2)+C\sum_i \xi_i{}^2 \qquad (2)$$

s. t. $\forall i,\ y_i(\boldsymbol{w}\cdot\boldsymbol{x}_\mathrm{i}+b)+\xi_i=1$

The main difference between standard SVM (1) and proximal SVM (2) is the constraints. Standard SVM employs an inequality constraint whereas proximal SVM employs an equality constraint. The intuition of Proximal SVM is shown in Figure 2. We can see that standard SVM only considers points on the wrong side of $\boldsymbol{w}\cdot\boldsymbol{x}_i+b=1$ and $\boldsymbol{w}\cdot\boldsymbol{x}_i+b=-1$ as training errors. However, in proximal SVM, all the points not located on the two planes are treated as training errors. In this case the value of training error ξ_i in (2) may be positive or negative. The second part of the objective function in (2) uses a squared loss function $\sum_i \xi_i{}^2$ instead of $\sum_i \xi_i$ to capture this new notion of error.

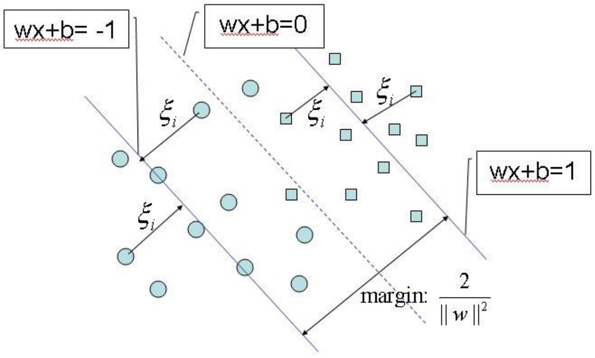

Figure 2. Proximal SVM

The proximal SVM made these modifications mainly for efficiency consideration. [3] proposed an algorithm to solve (2) using KKT conditions and Sherman-Morrison-Woodbury formula. This algorithm is very fast and has comparable effectiveness with standard SVM when the data dimension is far less than the number of training data (*n* << *m*). However, in text classification *n* usually has the same magnitude with *m* and the condition *n* << *m* is not hold anymore. To the best of our knowledge, little research works has been conducted to show the performance of proximal SVM with high dimensional data.

Although the original PSVM algorithm of [3] is not suitable for high dimensional data, Formula (2) can be solved efficiently for high dimensional data using iterative methods. We have applied the proximal SVM for text classification but found that when the data are unbalanced, i.e. when the amount of positive data are much more than negative data, or vice versa, the effectiveness of proximal SVM deteriorates more quickly than standard SVM. Data unbalance is common in text classification, which motivates us to search for an extension to proximal SVM to deal with this problem.

3. Weighted proximal SVM Model

We show the reason why the original proximal SVM is not suitable for classifying unbalanced data in this section. To the unbalanced data, without lose of generality, suppose the amount of positive data is much fewer than the negative data. In this case the total accumulative errors of negative data are much higher than that of positive data. Consequently, the bounding plane $\boldsymbol{w}\cdot\boldsymbol{x}_i+b=1$ will shift towards the direction opposite to the negative data to produce a larger margin at the price of increasing the positive errors. Since the positive data are rare, this action will lower the value of objective function (2). Then the separating plane will be biased to the positive data and result in a higher precision and a lower recall for the positive training data.

To solve this problem, we assign a non-negative weight δ_i to each training error ξ_i and convert the optimization problem (2) to the following form:

$$\min \frac{1}{2}\nu(\|\boldsymbol{w}\|^2+b^2)+\frac{1}{2}\sum_i \delta_i{}^2\xi_i{}^2 \qquad (3)$$

s. t. $\forall i,\ y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)+\xi_i=1$

The differences between (2) and (3) are:

1. Formula (2) assumes all the training errors ξ_i are equally weighted, but in Formula (3) we use a non-

negative parameter δ_i to represent the weight of each training error ξ_i.

2. In Formula (3), we let *v*=1/(2C) and move the tradeoff parameter *C* from $\sum_i \xi_i^2$ to $(\|\boldsymbol{w}\|^2+b^2)$. The purpose of this movement is for notation simplicity in the later development of our solving method.

Though (3) can be solved using KKT conditions and Sherman-Morrison-Woodbury formula as showed in [3], this solving strategy is inefficient for high dimensional data like text documents. Instead, we convert (3) to an unconstrained optimization problem that can be directly solved using iterative methods.

The constraint of (3) can be written as:

$$\xi_i^2=(1-y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b))^2=(y_i-(\boldsymbol{w}\cdot\boldsymbol{x}_i+b))^2 \qquad (4)$$

Using (4) to substitute ξ_i in the objective function of (3), we get an unconstrained optimal problem:

$$\min\ f(\boldsymbol{w},b)=\frac{1}{2}v(\|\boldsymbol{w}\|^2+b^2)+\frac{1}{2}\sum_i\delta_i^2(y_i-(\boldsymbol{w}\cdot\boldsymbol{x}_i+b))^2 \quad (5)$$

For notation simplicity, let $X\in R^{m\times n}$ denote the TF*IDF matrix of documents whose row vectors are $\mathbf{x}_i$. Suppose $\boldsymbol{e}$ is a vector whose elements are all 1. Let $A=[X,\ \boldsymbol{e}]\in R^{m\times(n+1)}$, $\boldsymbol{\beta}=[\boldsymbol{w},b]\in R^{(n+1)}$ and let $\Delta\in R^{m\times m}$ denotes a diagonal matrix whose non-zero elements are $\Delta_{ii}=\delta_i$, then (5) can be written as:

$$\min\ f(\boldsymbol{\beta})=\frac{1}{2}v\|\boldsymbol{\beta}\|^2+\frac{1}{2}\|\Delta(\mathbf{y}-\mathrm{A}\boldsymbol{\beta})\|^2 \qquad （6）$$

The gradient of $f(\boldsymbol{\beta})$ is:

$$\nabla f(\boldsymbol{\beta})=v\boldsymbol{\beta}-(\Delta\mathrm{A})^{\mathrm{T}}(\Delta\mathrm{y}\text{-}\Delta\mathrm{A}\boldsymbol{\beta})$$

$$=(v\mathbf{I}+(\Delta\mathrm{A})^{\mathrm{T}}(\Delta\mathrm{A}))\boldsymbol{\beta}-(\Delta\mathrm{A})^{\mathrm{T}}(\Delta\mathbf{y})$$

The Hessian matrix of $f(\boldsymbol{\beta})$ is:

$$\mathrm{H}=v\mathbf{I}+(\Delta\mathrm{A})^{\mathrm{T}}(\Delta\mathrm{A})$$

From v>0 and the elements of Δ and *A* are non-negative, it is easy to prove H is positive definite. The solution of (6) is found when $\nabla f(\beta)$=0, that is:

$$(v\mathbf{I}+(\Delta\mathrm{A})^{\mathrm{T}}(\Delta\mathrm{A}))\boldsymbol{\beta}=(\Delta\mathrm{A})^{\mathrm{T}}(\Delta\mathbf{y}) \qquad (7)$$

Equation (7) can be generally written as (shift*I + A'A)x=A'b, where A is a high dimensional sparse matrix. The CGLS /LSQR [9] algorithm is dedicated to efficiently solve this problem.

4. Algorithm Design

There are two main concerns in the algorithm design: how to set the parameters and how to solve Equation (7) efficiently. We will address these concerns in this section.

4.1. Parameter Tuning

Several parameters need to be decided in the training algorithm. Parameter *v* controls the tradeoff between maximizing the margin and minimizing the training errors. Parameters $\delta_i, i=1,2,\ldots,m$ control the relative error weights of each training example. To simplify the parameter setting for unbalanced data problem, we set the error weight of all positive training data to δ_+ and all negative training data to δ_-. Then we only need to set three parameters: *v*, δ_+ and δ_-. These parameters can be decided by statistical estimation methods on the training data, such as LOO (Leave-One-Out cross-validation), k-fold cross validation, etc. If we iteratively update the weights by the separating plane obtained from previous round of training, we essentially obtain a boosting based method such as AdaBoost [13]. However, a disadvantage of using these boosting based and cross-validation based methods is that they need too much training time for parameter estimation.

To obtain a more efficient method than the boosting based methods, we have developed a simple method that can estimate the parameters based on the training data. It can achieve comparable effectiveness as compared to algorithms that using standard SVM plus cross validation techniques. Our parameter estimation method is as follows.

To get a balanced accumulative error on both positive and negative data, it is better to have the following condition:

$$\sum_{y_i=1}\delta_+^2\xi_i^2=\sum_{y_i=-1}\delta_-^2\xi_i^2$$

If we assume the error ξ_i of both positive and negative training data has the same expectation, we can get:

$$N_+\delta_+^2=\delta_-^2N_- \qquad (8)$$

where *N*+ is the number of positive training examples and *N*- is the number of negative training examples. Then we set the parameter δ_- and δ_+ as follows.

Set δ_-=1

Set ratio=$\sqrt{N_-/N_+}$

Set δ_+=1+ (ratio-1)/2

Notice that we do not set δ_+=ratio to exactly satisfy Equation (8). Instead, we use a conservative setting strategy to make the precision of a minor class a little higher than recall. This strategy usually results in higher accuracy for unbalanced data.

Parameter v is set as follows.

$v = 2 * average(\delta_i \| \boldsymbol{x}_i \|)$

When the data are exactly balanced (the number of positive examples is equal to the number of negative examples), this method will result in $\delta_- = \delta_+ = 1$ and make WPSVM equal to PSVM. Therefore, PSVM can be viewed as a special case of WPSVM.

To give an intuitive example of the differences between WPSVM and PSVM, we manually generated a balanced data set and an unbalanced dataset in a two dimensional space. Then we calculated the separating plane of WPSVM and PSVM respectively. The results are shown in Figure 3 and Figure 4.

Figure 3 shows that the separating planes for PSVM and WPSVM are almost the same when the data are balanced. Figure 4 shows when the data is unbalanced, the separating plane for WPSVM resides in the middle of the positive and negative examples, but the separating plane for PSVM is inclined to the positive examples.

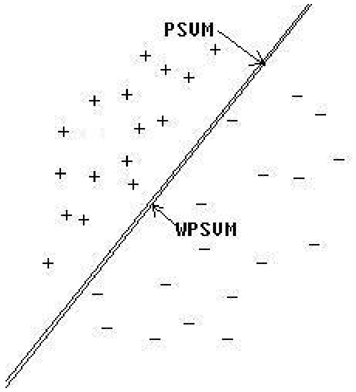

Figure 3. Separating planes for balanced data

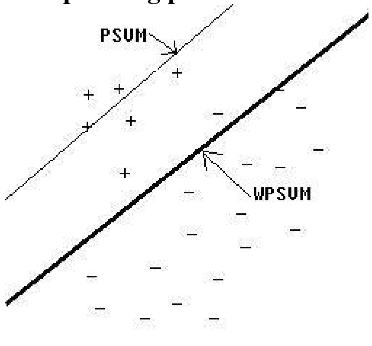

Figure 4. Separating planes for unbalanced data

4.2. Training Algorithms

We tried several methods to solve equation (7) and found CGLS [9] has the best performance. However, many other iterative optimal methods can also be used to solve Equation (7).

The complexity of the training algorithm is dominated by the algorithm used for solving Equation (7). Usually this kind of algorithms has *O(KZ)* time complexity and *O(Z)* space complexity where *K* is the number of iterations and *Z* is the number of non-zero elements in the training vectors.

Iterative method can only find an approximate solution to the problem. The more the number of iterations is used, the longer the training time is required and the iterative solution is closer to the optimal solution. However, when the iteration count archives a certain number, the classification result will not change when the number of iterations continues to increase. Therefore it is important to select a good terminating condition to obtain a better tradeoff between training time and classification accuracy. Since the number of required iterations may vary for different dataset, we make the terminating condition as an adjustable parameter when implementing the WPSVM algorithm.

5. Experiments

Rationale:

Our experiments evaluate the relative merits of WPSVM and other SVM based methods. We will verify the following hypotheses for text datasets:

1. WPSVM (with default parameter settings) has the same classification power as standard SVM plus cross-validation, has slightly better classification power than standard SVM (with default parameter settings) and has much better classification power than PSVM
2. WPSVM is much more efficient than standard SVM

Data sets:

The dataset that we choose is a textual dataset RCV1-v2 [8]. RCV1 (Reuters Corpus Volume I) is an archive of over 800,000 manually categorized newswire stories recently made available by Reuters, Ltd. for research purposes. Lewis, et al [8] made some corrections to the RCV1 dataset and the resulting new dataset is called RCV1-v2.

The RCV1-v2 dataset contains a total of 804,414 documents. The benchmark results of SVM, weighted k-NN and Rocchio-style algorithms on RCV1-v2 are reported in [8]. The results show that SVM is the best method on this dataset. To make our experimental results comparable with the benchmark results, we strictly follow the instruction of [8]. That is, we use the same vector files, training/test split and effective measures as in [8].

Text data representation:

The feature vector for a document was produced from the concatenation of text in the <headline> and <text> tags. After tokenization, stemming and stopword removal. 47,219 terms that appears in the training data are used as features. The features are weighted using the TF*IDF indexing schema and then being cosine normalized. The resulting vectors are published at [7]. We directly use these vectors for our experiments.

Training/test split:

The training/test split is according to the publishing time of the documents. Documents published from August 20, 1996 to August 31, 1996 are treated as training data. Documents published from September 1, 1996 to August 19, 1997 are treated as test data. This split produces 23,149 training documents and 781,256 test documents.

Categories and Effective measures:

Each document can be assigned labels according to three different category sets: Topics, Industries or Regions. For each single category, the one-to-rest strategy is used in the experiments. In other words, when classifying category X, all the examples labeled X are defined as positive examples, and the other examples are defined as negative examples.

The F1 measure is used to evaluate the classification quality of different methods. F1 is determined by Precision and Recall. The Precision, Recall, and F1 measures for a single category are defined as follows.

$$\text{Precision}=\frac{\text{\# of correctly classified positive examples}}{\text{\# of classifier predicted positive examples}}$$

$$\text{Recall} = \frac{\text{\# of correctly classified positive examples}}{\text{\# of real positive examples}}$$

F1 = (2*Precision*Recall) / (Precision + Recall)

The average effectiveness is measured by the average micro-F1 and average macro-F1. Average macro-F1 is the average value of each single F1 in the category set. Average micro-F1 is defined as follows.

$$\text{microP}=\frac{\sum_{i}\text{\# of correctly predicted docs for category i}}{\sum_{i}\text{\# of docs that are predicted as category i}}$$

$$\text{microR}=\frac{\sum_{i}\text{\# of correctly predicted docs for category i}}{\sum_{i}\text{\# of docs that truely belong to category i}}$$

Ave micro-F1=(2*microP*microR)/(microP+microR)

5.1. Experiments on WPSVM's Effectiveness

In the effectiveness testing experiments, we compare the F1 measure on the following:

WPSVM: Our proposed algorithm, using the parameter estimating method presented in section 4.1.

PSVM: Set all δ_i in WPSVM model equal to 1 and make it equivalent to the proximal SVM algorithm.

SVM light: Using SVM light v 6.01 [5] with default parameter settings.

SVM.1: This algorithm is a standard SVM plus threshold adjustment. It is a benchmark method used in [8]. In this algorithm, SVM light was run using default parameter settings and was used to produce the score. The threshold was calculated by the SCutFBR.1 [12] algorithm.

SVM.2: This algorithm is a standard SVM plus LOO cross validation. It was first introduced in [6] and named as SVM.2 in [8]. In this algorithm, SVM light was run multiple times with deferent –j parameters and the best –j parameter was selected by LOO validation. The -j parameter controls the relative weighting of positive to negative examples. This approach solved the data unbalance situation by selecting the best –j parameter. The experiments were separately performed on each category using the one-to-rest strategy. The dataset scale for each category is shown in table 1.

Table 1. Dataset scale for each category

Number of training examples	23149
Number of test examples	781256
Number of features	47219
Average Number of non-zero elements	123.9

We first introduce the results on the Topics categories. There are total 101 Topics categories that at least one positive example appears in the training data. We calculate the F1 value for the five algorithms on each category (The F1 value of SVM.1 and SVM.2 is calculated by the contingency table published at [7]). Figure 5 shows the changes of F1 value from unbalanced data to balanced data for the five algorithms. Categories are sorted by training set frequency, which is shown on the x-axis. The F1 value for a category with frequency x has been smoothed by replacing it with the output of a local linear regression over the interval x−200 to x+200.

From the results we can see that when the training data is relatively balanced (the right part Figure 5), the F1 measure for the five algorithms has no big differences. When the training data is unbalanced (the left part of Figure 5), the classification quality of WPSVM is between SVM.1 and SVM.2. Both have better classification quality than SVM light and PSVM. Figure 5 also shows the classification quality of PSVM deteriorates more quickly than that of SVM light when the data become unbalanced.

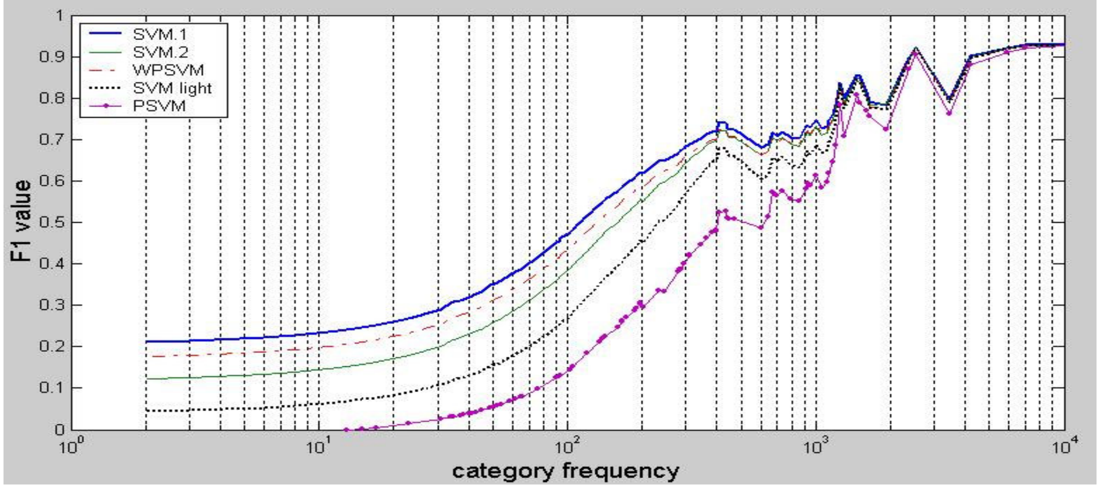

Figure 5. F1 measure for five methods on 101 Topic categories

Table 2 shows the average F1 measure of the 101 categories. The results of SVM.1 and SVM.2 are the values reported in [8]. It can be seen that the overall performance of WPSVM, SVM.1 and SVM.2 are better than that of SVM light and PSVM. SVM.1 has the best average effectiveness, especially in average macro-F1. This is mainly because when the training data are extremely unbalanced (e.g. the positive ratio is less than 0.1%), the threshold adjustment method is better than both WPSVM and SVM.2.

Table 2. Average F1 measure for Topics

Algorithms	Average micro-F1	Average macro-F1
PSVM	0.767	0.354
SVM light	0.804	0.472
WPSVM	0.808	0.589
SVM.2	0.810	0.557
SVM.1	0.816	0.619

Table 3. Average F1 for Industries and Regions

Algorithms		Average micro-F1	Average macro-F1
Industries (313)	SVM.1	0.513	0.297
	WPSVM	0.520	0.301
Regions (228)	SVM.1	0.874	0.601
	WPSVM	0.862	0.558

We also test the effectiveness of WPSVM on the 313 Industries categories and 228 Regions categories. The average F1 measures of these categories are shown in Table 3. The results of SVM.1 shown in table 3 are the values reported in [8]. We can see that in the Industries and Regions Split, the effectiveness of WPSVM is also comparable with SVM.1.

The effectiveness experiments show the overall classification quality of WPSVM is comparable with SVM.1 and SVM.2, which are the best methods of [8], and is better than SVM light and PSVM. However, SVM.1 and SVM.2 require training many times to estimate a good parameter whereas WPSVM only require training once.

5.2. Experiments on Computational Efficiency

The computational efficiency is measured by the actual training time and memory usage respectively. Since SVM.1 and SVM.2 require running SVM light many times, their efficiency must be less than SVM light. Thus in the experiments, we only compare the efficiency of WPSVM and SVM light. We run each algorithm on 5 training dataset with different size. The vector files of [8] are published as one training file and 4 test files. We use the training file as the first dataset and then incrementally append the remaining four test files to form the other four datasets. The number of training examples for the 5 datasets is 23149, 222477, 421816, 621392 and 804414 respectively. The training time is measured in second. Both algorithms ran on an Intel Pentium 4 Xeon 3.06G computer.

We found that when using SVM light for the same training size, balanced data required more training time than the unbalanced data. Thus, we did two groups of efficiency experiments. One group uses category CCAT as positive examples. The ratio of CCAT is 47.4% and it makes this group as a balanced example. The other group is an unbalanced example. It uses GDIP as positive examples. The ratio of GDIP is 4.7%.

Table 4 shows the training time of WPSVM and SVM light V6.01 on the two groups. We can see that the training time of WPSVM is far less than the training time of SVM light and is not affected by the data unbalanced-ness problem.

1550-4786/05 $20.00 © 2005 IEEE

Table 4. Training time comparison

No. of training data	CCAT		GDIP	
	WPSVM	SVM light	WPSVM	SVM light
23149	1.6	43	2.1	9.1
222477	42.7	1313	35	317
421816	80.5	3306	100	884
621392	194.4	5110	171	1599
804414	273.4	10986	276	2458

The memory usage required for both WPSVM and SVM light is determined by the training size, regardless of whether the data are balanced or unbalanced. Figure 6 shows the memory requirements of the two algorithms with different training sizes. We can see that the memory requirement of WPSVM is slightly less than SVM light. This is because WPSVM almost only require the memory to store the training data but SVM light requires additional working space.

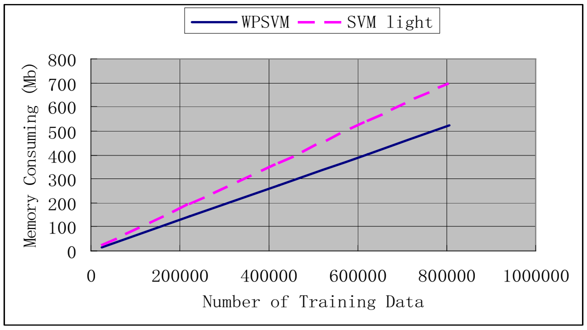

Figure 6. Memory consume comparison

6. Conclusion and Future work

In this paper, we proposed a weighted proximal SVM model, which assigns a weight to each training error. We successfully applied the WPSVM model to text classification problem by a simple parameter estimation method and an algorithm for solving the equations directly instead of using KKT conditions and the Sherman-Morrison-Woodbury formula. The experiments showed that our proposed method can achieve comparable classification quality as the standard SVM when supplemented with validation techniques, but is more computationally efficient than the standard SVM. We only validated the effectiveness of our algorithm on text classification in this paper. As a general linear SVM classification algorithm, it can also be used in other classification tasks. It is worth pointing out that in this paper we only demonstrated the advantage of WPSVM in solving the data unbalanced-ness problem. However the WPSVM model may have other potential use. In WPSVM, the relative importance of each training point can be adjusted based on other prior knowledge.

7. Acknowledgement

Qiang Yang is supported by a grant from Hong Kong RGC: HKUST6187/04E.

8. References

[1] Baeza-Yates, R. and Ribeiro-Neto, B., *Modern Information Retrieval*. Addison Wesley, 1999.

[2] Burges, C., *A Tutorial on Support Vector Machine for Pattern Recognition*. Data Mining and Knowledge Discovery, 1998.

[3] Fung, G. and Mangasarian, O. L., *proximal Support Vector Machine Classifiers*. In Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001), 2001.

[4] Joachims, T., *Making Large-Scale SVM Learning Practical.* Advances in Kernel Methods – Support Vector Learning, 1999

[5] Joachims T., SVM Light: Support Vector Machine. Feb 9th, 2004. http://svmlight.joachims.org.

[6] Lewis, D. D., *Applying support vector machines to the TREC-2001 batch filtering and routing tasks*. In The Tenth Text REtrieval Conference (TREC 2001), pages 286–292, Gaithersburg, MD 20899-0001, 2002. National Institute of Standards and Technology.

[7] Lewis, D. D., *RCV1-v2/LYRL2004: The LYRL2004 Distribution of the RCV1-v2 Text Categorization Test Collection (12-Apr-2004 Version)*. http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

[8] Lewis, D. D., Yang, Y. Rose, T. and Li, F., *RCV1: A New Benchmark Collection for Text Categorization Research*. Journal of Machine Learning Research, 5:361-397, 2004.

[9] Paige C. C. and Saunders, M. A., *Algorithm 583; LSQR: Sparse linear equations and least-squares problems.* TOMS 8(2), 195--209, 1982.

[10] Platt, J., *Fast Training of Support Vector Machines using Sequential Minimal Optimization.* Advances in Kernel Methods – Support Vector Learning, 1998

[11] Vapnik, V. N., *Statistical Learning Theory*. John Wiley & Sons, 1998

[12] Yang Y., *A study on thresholding strategies for text categorization*. In the Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 01), 2001.

[13] Freund, Y. and Schapire, R, *Experiments with a New Boosting Algorithm*. Machine Learing: Proceedings of the Thirteenth International Conference (ICML 96), 199