Text Classification from Positive and Unlabeled Documents

Hwanjo Yu
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
USA
hwanjoyu@uiuc.edu

ChengXiang Zhai
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
USA
czhai@cs.uiuc.edu

Jiawei Han
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
USA
hanj@cs.uiuc.edu

ABSTRACT

Most existing studies of text classification assume that the training data are *completely* labeled. In reality, however, many information retrieval problems can be more accurately described as learning a binary classifier from a set of *incompletely* labeled examples, where we typically have a small number of labeled *positive* examples and a very large number of unlabeled examples. In this paper, we study such a problem of performing Text Classification WithOut labeled Negative data (*TC-WON*). In this paper, we explore an efficient extension of the standard Support Vector Machine (SVM) approach, called SVMC (Support Vector Mapping Convergence) [17], for the TC-WON tasks. Our analyses show that when the positive training data is not too under-sampled, SVMC significantly outperforms other methods because SVMC basically exploits the natural "gap" between positive and negative documents in the feature space, which eventually corresponds to improving the generalization performance. In the text domain there are likely to exist many gaps in the feature space because a document is usually mapped to a sparse and high dimensional feature space. However, as the number of positive training data decreases, the boundary of SVMC starts overfitting at some point and end up generating very poor results. This is because when the positive training data is too few, the boundary over-iterates and trespasses the natural gaps between positive and negative class in the feature space and thus ends up fitting tightly around the few positive training data.

Categories and Subject Descriptors

I.7.m [**Computing Methdologies**]: Document and Text Processing—*Miscellaneous*

General Terms

Algorithms, Performance

Keywords

Text Classification, Text Filtering, SVM, Machine Learning

CIKM'03, November 3–8, 2003, New Orleans, Louisiana, USA.

1. INTRODUCTION

Binary classification of text is a fundamental problem in information retrieval, and occurs in several different forms. For example, in text categorization, a collection of binary classifiers (one for each category) are often used to classify a document into one or more categories. In information filtering, identifying documents that are interesting to a user is essentially an online binary classification problem. Even in ad hoc retrieval, the goal is also to classify documents as either "relevant" or "non-relevant", though a ranking of documents is often presented to the user; the connection of ad hoc retrieval with binary classification is amplified in relevance-feedback, where the task can be formulated as learning a binary classifier based on an extremely small number of positive examples.

In most scenarios mentioned above, the binary classification problem involves learning an *unbalanced* classifier, where the number of documents in each of the two categories is extremely unbalanced, with the interesting or "positive" category having generally very few documents, while the other or "negative" category having far many more documents. Moreover, we often do not have complete judgments of positive documents in our set of training documents, i.e., many unlabeled documents may actually be positive. In general, this may be because of the labor required to label all examples, but in some cases, the task setup makes it impossible to obtain complete judgments of positive documents. One such a case is adaptive information filtering, where the system can only hope to obtain judgments on any documents that the system has decided to send to a user; it is impossible to obtain judgments on any of the "discarded" documents, in which the user will not have any chance to see. This suggests the following interesting formulation of the binary classification problem, which, we believe, represents a more realistic setup of the problem in many cases:

- The positive class (e.g., the category of relevant documents) is a coherent subset of all data, and the negative class is "everything else".
- In any random sample of documents, the number of positive documents and that of negative ones are seriously unbalanced, with the former significantly smaller than the latter.
- The training data is composed of a (small) set of labeled *positive* documents and a (very large) set of unlabeled documents.

Although text classification has been extensively studied, most existing studies, with only a few exceptions [10, 9], assume that the training examples are completely labeled. It appears that little

work has been done to address the binary classification problem as formulated above, which is arguably more realistic.

Our problem is essentially Text Classification WithOut labeled Negative data (*TC-WON*). It can also be considered as the problem of *text classification with noisy negatives* since the unlabeled documents can be thought of as a good approximation of negative documents with some noise. From the viewpoint of machine learning, the challenge is to learn from a set of unlabeled documents with only some positive documents explicitly labeled. Indeed, the lack of reliably judged negative documents may significantly degrade the classification performance. This is true especially for those discriminative models such as the Support Vector Machines (SVMs), which are by far among the most effective text classification methods, since the inevitable noise (i.e,. unlabeled positive documents) in the negative example set is often close to the decision boundary, and it would push the decision boundary away from the optimal and toward positive documents undesirably.

In this paper, we explore an efficient extension of the standard Support Vector Machine (SVM) approach, called SVMC (Support Vector Mapping Convergence) [17], for the TC-WON tasks. Our analyses show that when the positive training data is not too undersampled, SVMC significantly outperforms other methods because SVMC basically exploits the natural "gap" between positive and negative documents in the feature space, which eventually corresponds to improving the generalization performance. In the text domain there are likely to exist many gaps in the feature space because a document is usually mapped to a sparse and high dimensional feature space. However, as the number of positive training data decreases, the boundary of SVMC starts overfitting at some point and end up generating very poor results. This is because when the positive training data is too few, the boundary over-iterates and trespasses the natural gaps between positive and negative class in the feature space and thus ends up fitting tightly around the few positive training data. Section 3 analyze this further and Section 4.4.3 shows an example of the "over-iteration" problem.

In the rest of the paper, we first briefly review existing work on applying SVMs to text classification in Section 2. We then analyze the Mapping Convergence (MC) and the SVMC algorithm in Section 3. The experiments and results are discussed in Section 4. Finally, we summarize our work and discuss future work in Section 5.

2. SVMS FOR TEXT CLASSIFICATION

Machine learning has been applied to text classification extensively and has enjoyed a great deal of success [6, 3, 16, 15, 1, 12]. Among the many learning algorithms, Support Vector Machine (SVM)[14] appears to be most promising. Indeed, since Joachims's pioneering work on applying SVMs to text classification [3], a number of studies have shown that SVMs outperform many other text classification methods (e.g., [16, 3, 4]). In [5], a theoretical learning model of text classification for SVMs is developed, which provides some explanation for why and when SVMs perform well for text classification. In particular, it suggests that text classification tasks are generally linearly separable with a large margin, which, along with a low training error, can lead to a low expected prediction error.

Scholkopf et al. (1999) suggested a method of adapting the standard SVM to the so-called one-class classification problem, where only positive examples are available for learning. With a strong mathematical foundation of SVM, the One-class SVM (OSVM) computes a boundary of positive data without negative data in the feature space [13, 8]. OSVM has two parameters – ν (to control the noise in the training data) and γ (to control the "smoothness" of the boundary). OSVM has the same advantages as SVM such as efficient handling of sparse feature spaces and nonlinear classification using advanced kernel functions. However, OSVM requires a much larger amount of positive training data to induce an accurate class boundary because its support vectors (SVs) of the boundary come only from the positive data set and the small number of positive SVs can hardly cover the major directions of the boundary, especially in high dimensional spaces. As a result, OSVM tends to overfit and underfit easily. Tax [13] proposed a sophisticated method which uses artifically generated unlabeled data to optimize the OSVM's parameters in order to "balance" between overfitting and underfitting. However, even with the best parameter setting, its performance is still significantly worse than the original SVM with negative data due to the short of SVs to describe the boundary "completely". Moreover, the proposed optimization method in [13] is infeasibly inefficient in high dimensional spaces.

Positive Example-Based Learning (PEBL) framework has been proposed for Web page classification without labeled negative examples in [18]. PEBL runs SVM iteratively to induce a SVM-based boundary around positive data set. However, this method is limited to binary features due to the usage of monotone disjunction learning at the initial stage, which is appropriate for Web page classifications, where features about structural information (e.g., links, url, tags, headings) are also often very useful. Another problem is that the training time of PEBL highly depends on the number of documents, and can be significantly more complex than the standard SVM, whose training time is already at least quadratic to the size of training data. This problem becomes more critical when the size of unlabeled data set is very large.

Support Vector Mapping Convergence (SVMC), which extends the PEBL approach to handle continuous features (e.g., TF-IDF value) and improves the efficiency of PEBL, has shown superior performance over other methods on letter recognition, breast cancer classification, and Web page data sets [17]. However, our extensive analyses disclose that such method has an "over-iteration" problem and thus needs further improvement for document filtering or classification purpose.

3. MAPPING CONVERGENCE (MC)

In this section, we analyze the Mapping-Convergence (MC) algorithm, which is the basis of the SVMC algorithm, to disclose the previously hidden properties of the MC and SVMC algorithms. For convenience of presentation, we use the following notations throughout this paper.

- $\mathcal{U}$ is the feature space for the universal set such that $\mathcal{U} \subseteq \Re^m$ where m is the number of dimensions.
- U (unlabeled data set) is a uniform sample of the universal set.
- x is a data instance such that $x \in \mathcal{U}$.
- $\mathcal{P}$ is a subspace for positive class within $\mathcal{U}$, from which positive data set P is sampled.

For an example of Web page classification, the universal set is the entire Web, U is a sample of the Web, P is a collection of Web pages of interest, and $x \in \Re^m$ is an instance of Web page.

3.1 Negative Strength

Here we introduce the notion of "negative strength", which is key to the MC algorithm.

Input: - positive data set P, unlabeled data set U
Output: - a boundary function h_i

Ψ_1: an algorithm identifying "strong negatives" from U
e.g., Rocchio or OSVM
Ψ_2: a supervised learning algorithm that maximizes the margin
e.g., SVM

Algorithm:
1. Use Ψ_1 to construct a classifier h_0 from P and U which classifies only "strong negatives" as negative and the others as positive
2. Classify U by h_0
   * $\hat{N}_0$:= examples from U classified as negative by h_0
   * $\hat{P}_0$:= examples from U classified as positive by h_0
3. Set $N := \emptyset$ and $i := 0$
4. Do loop
   4.1. $N := N \cup \hat{N}_i$
   4.2. Use Ψ_2 to construct h_{i+1} from P and N
   4.3. Classify $\hat{P}_i$ by h_{i+1}
      * $\hat{N}_{i+1}$:= examples from $\hat{P}_i$ classified as negative by h_{i+1}
      * $\hat{P}_{i+1}$:= examples from $\hat{P}_i$ classified as positive by h_{i+1}
   4.4. $i := i + 1$
   4.5. Repeat until $\hat{N}_i = \emptyset$
5. return h_i

Figure 1: MC algorithm

Let $h(x)$ be the boundary function of the positive class in $\mathcal{U}$, which outputs the distance from the boundary to the instance x in $\mathcal{U}$ such that

$$\begin{aligned} h(x) > 0 \quad & if\ x\ is\ a\ positive\ instance, \\ h(x) < 0 \quad & if\ x\ is\ a\ negative\ instance, \\ |h(x)| > |h(x')| \quad & if\ x\ is\ located\ farther\ than\ x' \\ & from\ the\ boundary\ in\ \mathcal{U}. \end{aligned}$$

DEFINITION 1 (STRENGTH OF NEGATIVE INSTANCES [17]). *For two negative instances x and x' such that $h(x) < 0$ and $h(x') < 0$, if $|h(x)| > |h(x')|$, then x is **stronger** than x'.*

EXAMPLE 1. *Consider a resume page classification function $h(x)$ from the Web ($\mathcal{U}$). Suppose there are two negative data objects x and x' (non-resume pages) in $\mathcal{U}$ such that $h(x) < 0$ and $h(x') < 0$: x is "how to write a resume" page, and x' is "how to write an article" page. In $\mathcal{U}$, x' is considered more distant from the boundary of the resume class because x has more features relevant to the resume class (e.g., the word "resume" in text) though it is not a true resume page.*

3.2 MC Analysis

The MC algorithm is composed of two stages: the *mapping stage* and the *convergence stage*. In the mapping stage, the algorithm uses a weak classifier Ψ_1 (e.g., Rocchio or OSVM), which draws an initial approximation of "strong negatives" – the negative data located far from the boundary of the positive class in $\mathcal{U}$ (steps 1 and 2 in Figure 1). Based on the initial approximation, the convergence stage runs in iteration using a second base classifier Ψ_2 (e.g., SVM), which maximizes margin to make a progressively better approximation of negative data (steps 3 through 5 in Figure 1). Thus the class boundary eventually converges to the boundary around the positive data set in the feature space.

The MC algorithm is described in Figure 1. To illustrate the MC process, consider an example of data distribution in one dimensional feature space in Figure 2, in which U is composed of eight data clusters, and the fifth one from left is positive and the rest are negative. If U is completely labeled, a margin-maximization algorithm such as SVM trained from U would generate the optimal boundary (b_d, b'_d), where b_d maximizes the near and farther points of the gap (g_n and g_f) between positive and negative clusters. Unfortunately, under the common scenarios of SCC, the only labeled data is assumed to be the dark center within the positive cluster, and the rest are unlabeled.

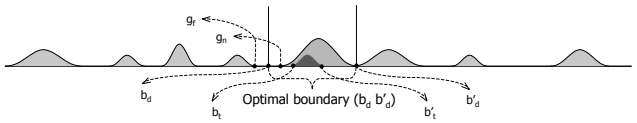
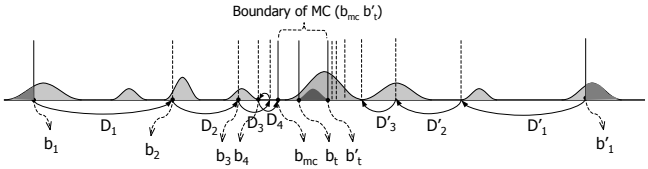
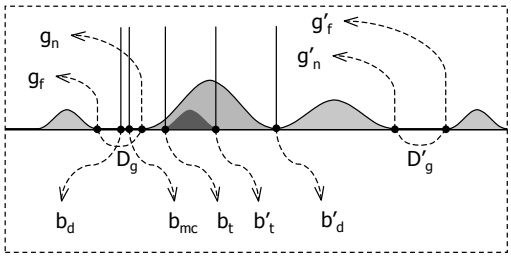


Figure 2: Example of data distribution in one dimensional feature space. The fifth data cluster from left is positive and the rest are negative.



(a) MC process.



(b) Zoomed-in center.

Figure 3: Example of the MC algorithm in the one dimensional feature space.

Figure 3(a) illustrates the MC process given the labeled positive data P (i.e., the dark center) with the unlabeled U. Let the strong negatives that algorithm Ψ_1 identifies be the data outside of b_1 and b'_1 to left and right side respectively, which are located far from the positive data cluster in the feature space. (We will justify the validity of algorithm Ψ_1 later in this section.) At each iteration of Step 4 in Figure 1, algorithm Ψ_2 includes the data within D_i into N as shown in Figure 3(a), where i is the number of iterations and b_i is the starting point of ith iteration in the feature space. Since Ψ_2 maximizes the margin at each iteration, D_1 has the half margin of that between b_1 and b_t, and D_{i+1} will have the half margin of D_i.

However, if there exists no data before the boundary after ith iteration, b_{i+1} will retract to the point that there exists a data within D_i, because the data within D_i merged into N will be the nearest

negative data points of the next boundary, and Ψ_2 maximized the margin between the nearest data points (i.e., support vectors). For instance, in Figure 3(a), b_4 retracts a little from the boundary after 3rd iteration.

If there exists no data at all within D_i, the convergence will stop because nothing will be accumulated into N and thus more iterations after that will not affect the boundary. For instance, in Figure 3(a), b_{mc} will be the final boundary of MC *to the left side* because there exists no data within D_4 and thus nothing will be accumulated into N from that point. As we see from Step 4.5 of Figure 1, the MC algorithm will stop the convergence when nothing is accumulated into N. The final boundary of MC *to the right side* in the figure tightly fits around P (i.e., b'_t) because there is no gap between positive and negative clusters and thus the convergence will not stop until MC will include every unlabeled data beyond P into N.

From the above example, we see that MC will stop the convergence and locate the boundary close to the optimum if there exists a large gap between positive and negative data clusters. (e.g., b_{mc} is close to b_d in Figure 3(b).) b_d equally divides the margin between g_f (i.e., farther point of the gap) and g_n (nearer point of the gap) whereas b_{mc} does between g_f and b_t. However, if there is no wide gap between those (e.g., the right side of Figure 3(a)) or the positive labeled data is very few, the MC boundary will "over-iterate" and end up converging into the tight boundary (e.g., b'_t in Figure 3(a)).

How wide gaps or how many positive labeled data are needed to avoid the "over-iteration" problem? To provide an intuitive answer, we assume that the feature space has only one dimension and negative data exist only to one side (e.g., the left side of Figure 3(a)). We denote $|x - y|$ for the margin between two points x and y. Let D_g be the gap between g_n and g_f where g_n and g_f are the two ending points of the gap respectively near and far from the positive class as shown in Figure 3(b). Then we have the following lemma.

LEMMA 1. *If MC stops the convergence at ith iteration, there must exists a gap D_g such that $|g_f - g_n| > |g_n - b_t|$ and $b_i = g_f$, where b_i is the starting point of ith iteration.*

PROOF. First, if MC stops the convergence at ith iteration, b_i will be the starting point of the gap (i.e., g_f), because if g_f is farther than b_i from b_t, b_i retracts to g_f, and if there exist data on b_i, MC will not stop the convergence and continue to accumulate the data to N such that $b_{i+1} = g_f$. Second, MC includes the half margin of $|b_i - b_t|$ from b_i at ith iteration since Ψ_2 maximizes the margin. If MC stops the convergence at ith iteration, there must exists a gap at least from b_i to the half point to b_t, i.e., $|g_f - g_n| > \frac{|g_f - b_t|}{2}$ since $b_i = g_f$. Then, $|g_f - g_n| > \frac{|g_f - b_t|}{2} = \frac{|g_f - g_n| + |g_n - b_t|}{2}$, Thus $|g_f - g_n| > |g_n - b_t|$ □

THEOREM 1. *MC locates the boundary within the first gap D_g it confronts such that $|g_f - g_n| > |g_n - b_t|$, during the iterations.*

PROOF. From Lemma 1, we know that if MC do not see a gap D_g such that $|g_f - g_n| > |g_n - b_t|$ during the iterations, it will not stop the convergence. Let us assume that MC confronts a gap of D_g such that $|g_f - g_n| > |g_n - b_t|$ at ith iteration. Then, MC tries to include the margin of $\frac{|b_i - b_t|}{2}$ from b_i. If b_i is farther from b_t than g_f, $b_{i+1} = g_f$. If g_f is farther from b_t than b_i, $b_i = g_f$. Since $|g_f - g_n| > |g_n - b_t|$, $\frac{|g_f - b_t|}{2} = \frac{|g_f - g_n| + |g_n - b_t|}{2} < |g_f - g_n|$ Thus, nothing will be added to N at ith iteration, and the iteration will stop then. □

From Theorem 1, we can deduce the following observations:

- MC is not likely to locate the boundary exorbitantly far from the boundary of P unless U is extremely sparse, because a larger gap ($|g_f - g_n|$) is needed to stop the convergence as g_n (i.e., the nearer ending point of the gap) is becoming farther from b_t (or $|g_n - b_t|$ increases).
- The more P is under-sampled, the larger gaps between positive and negative classes are needed to avoid the "over-iteration" problem, because $|g_n - b_t|$ would increase as P is more under-sampled.

From the above observations, we see that MC is likely to generate the boundary close to the optimum when P is not too much under-sampled and wide gaps exist between P and N in the feature space.

In text classification or filtering tasks, each document is typically mapped to a sparse and high dimensional space, and thus there are likely to exist visible gaps among different groups of data. However, for some filtering tasks where the positive training data is very few, the MC algorithm might not work well since the boundary will not fall into the gap. Section 4 provides the experimental evidences of such cases.

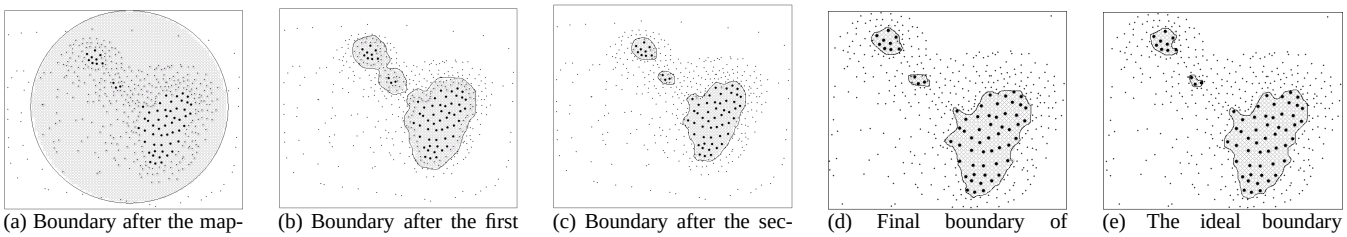
Note that the MC algorithm is quite general, as long as the component algorithms Ψ_1 and Ψ_2 satisfy the following requirements:

1. Ψ_1 *identifies strong negatives excluding false negatives.*

   Ψ_1 includes false negatives, they are likely to stay on the false negatives during the iterations which will deteriorate the boundary accuracy. We can use any reasonable classifier, such as Rocchio or OSVM, and adjust the threshold so that it makes near 100% recall by sacrificing precision. We use OSVM for Ψ_1 for our research, and set the bias b very low to achieve a high recall. The precision of Ψ_1 does not affect the accuracy of the final boundary as far as it approximates a certain amount of negative data because the final boundary will be determined by Ψ_2. Figure 4 shows an example of the boundary after each iteration of SVMC. The mapping stage only identifies very strong negatives by covering a wide area around the positives (Figure 4(a)). Although the precision quality of mapping is poor, the boundary at each iteration converges (Figures 4(b) and (c)), and the final boundary is very close to the ideal boundary drawn by SVM on P and *true* N (Figure 4(d) and 4(e)). Our experiments in Section 4 also show that the final boundary becomes very accurate although the initial boundary of the mapping stage is very rough by the "loose" setting of the threshold of Ψ_1.

2. Ψ_2 *must maximize margin.*

   SVM and Boosting are currently most popular supervised learning algorithms that maximize margin. We use SVM for Ψ_2 for our research. With a strong mathematical foundation, SVM automatically finds the optimal boundary without a validation process and without many parameters to tune. The small numbers of theoretically motivated parameters also work well with an intuitive setting. In practice, the soft constraint of SVM is necessary to cope with noise or outliers. However, in the SCC problem domains, P is unlikely to have a lot of noise since it is usually carefully labeled by users. Thus we can use a very low value for the soft margin parameter ν. In our experiments, a low setting (i.e., $\nu = 0.01$ or 0.1) of ν (the parameter to control the rate of noise in the training data) performed well for this reason. (When $\nu = 0$, it becomes the hard margin.) (We used ν-SVM for the semantically meaningful parameter [11, 2].)



(a) Boundary after the mapping stage (b) Boundary after the first iteration (c) Boundary after the second iteration (d) Final boundary of SVMC from P and U (e) The ideal boundary drawn by SVM from P and *true* N

Figure 4: Intermediate results of SVMC. P is big dots, and U is all dots (small and big).

3.3 Support Vector Mapping Convergence (SVMC)

The classification time of the final boundary of MC with $\Psi_2 = SVM$ is equal to that of SVM because the final boundary is a boundary function of Ψ_2. However, the training time of MC can be very long if $|U|$ is very large because the training time of SVM highly depends on the size of data set n ($\approx |U|$), and MC runs iteratively. Indeed, $t_{MC} = O(|U|^2 * log|U|)$ assuming *the number of iterations* $\approx log|U|$ and $t_{SVM} = O(|U|^2)$ where t_Ψ is the training time of a classifier Ψ. (t_{SVM} is known to be at least quadratic to n and linear to the number of dimensions; More discussion on the complexity of SVM can be found in [2].) However, decreasing the sampling density of U to reduce the training time could hurt the accuracy of the final boundary because the density of U will directly affect the quality of the SVs of the final boundary. Fortunately, it is possible to prevent the training time from increasing dramatically as the sample size grows, and this leads to the SVMC algorithm, which implements the MC algorithm with the training time being *independent* of the number of iterations and asymptotically equal to that of a SVM.

The basic idea of SVMC is to use a minimally required data set at each iteration such that the data set does not degrade the accuracy of the boundary. In this way, it saves the training time of each SVM maximally. To illustrate how SVMC achieves this, consider the point of starting the third iteration (when i = 2) in MC (Step 4.1 in Figure 1). After we merge $\hat{N}_2$ into N, in order to construct h_3, we may not need all the data from N since the data far from h_3 will not contribute to the SVs. The set of negative SVs of h_2 is the representative data set for $\hat{N}_0$ and $\hat{N}_1$, so we only keep the negative SVs of h_2 and the newly induced data set $\hat{N}_2$ to support the negative side of h_3.

CLAIM 1 (MINIMALLY REQUIRED NEGATIVE DATA [17]). *$\forall i \geq 1$, the minimally required negative data at $(i+1)$th iteration is $\hat{N}_i \cup$ negative SVs of h_i, which, along with P, can generate a h_{i+1} that is as accurate as one constructed from $\cup_{j=0}^{i}\hat{N}_j$ and P.*

Refer to [17] for the rationale for Claim 1.

However, Claim 1 only applies for the hard margin cases (i.e., when $\nu = 0$). In practice, due to the soft margin, the negative SVs of h_i may not totally represent $\cup_{j=0}^{i-1}\hat{N}_j$ especially in high dimensional spaces. In our experiments, MC performed a little better than SVMC for this reason.

For the minimally required data set for the *positive* side, we can never be sure whether we can exclude any data object from P at each iteration, because with the negative SVs different at each iteration, it is hard to predict which positive data will not be SVs independent of the number of iterations.

Surprisingly, adding the following statement between step 4.4 and 4.5 of the original MC algorithm of Figure 1 completes the SVMC algorithm.

```
Reset N with negative SVs of Ψ2
```

THEOREM 2 (TRAINING TIME OF SVMC [17]). *Suppose $t_{SVM} = n^2$, and $|\hat{N}_{i+1}| = \frac{|\hat{N}_i|}{2}$, $\forall i \geq 1$. Then, $t_{SVMC} \approx \frac{|U|^2}{3}$.*

Refer to [17] for the proof of Theorem 2.

Theorem 2 states that the training complexity of SVMC is asymptotically equal to that of SVM. Our experiments in Section 4 also show that SVMC trains much faster than MC with very close accuracy. As we show in Figure 4, under the certain conditions, the final boundary of SVMC becomes very close to the ideal boundary drawn by SVM with P and true N (Figure 4(d) and 4(e)).

4. EVALUATION

To study the effectiveness of the SVMC algorithm for TC-WON, we test it on two commonly used text categorization data sets. We compare it with the standard SVM trained by assuming all unlabeled examples as negative examples to see if the MC algorithm can effectively handle noise in negative examples. We also compare it with SMC to see how well SVMC improves the efficiency. Finally, we compare it with alternative approaches to learning from positive and unlabeled examples, including OSVM and S-EM [7].

4.1 Performance Measures

To evaluate a TC_WON system, we use the F_1 measure, which is a commonly used performance measure for text classification and also used in [7] – one of the most recent works on TC_WON. This measure combines precision and recall in the following way:

$$precision = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ preditions}$$

$$recall = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ examples}$$

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

We also report the classification accuracy, even though it is not a good measure for the TC_WON task because always predicting negative would give a high accuracy.

4.2 Data sets

We used the two commonly used corpora for text classification tasks – Reuters-21578[1] and Webkb[2].

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578

[2]http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

We used the ModApte version of the Reuters corpus, which has 9603 training documents and 3299 testing documents. Each document was represented as a stemmed, TFIDF weighted word frequency vector.[3] Each vector had unit modulus. A list of stop words were removed, and words occurring in less than three documents were also ignored. Using this representation, the document vectors had around 10000 dimensions.

The Webkb data set contains 8282 Web pages gathered from university computer science departments. Each web page was represeted as a binary feature vector. A feature (i.e., an element of a vector) is a predicate indicating whether each term or special character occurs in each part. (e.g., ' ' in URL, a word 'course' in title) We considered the most commonly used features for web pages such as URL, title, headings, links, anchor text, regular text, etc. We did not perform word stemming or a stoplist because it usually hurts performance in Web page classification. For example, a common stopword, "I" or "my", is a good indicator of a student homepage.

4.3 Experiment Methodology

To fully test the effectiveness and efficiency of SVMC, we compare seven methods – *SMC, SVMC, OSVM, SVM_NN, S-EM, NB_NN, ISVM*.

SMC is "Simple" MC described at the beginning of Section 3.3.

OSVM is described in Section 2.

SVM_NN is standard SVM trained using positive examples and unlabeled documents as a subsititute for negative documents. This is not unreasonable, as the unlabeled documents can be thought of as a good approximation of negative documents. However, the small number of false positives are likely to affect the support vectors (SVs) of the boundary, which hurts the recall performance, as shown in our experiments.

S-EM is a recent extension of the EM algorithm for text classification from positive and unlabeled documents [7]. Based on the EM algorithm, S-EM has the fundamental limitations of generative models, the attribute independent assumption which results in linear separation, and the requirement of good estimation of prior probabilities. This is confirmed in our experiments.

NB_NN (Naive Bayes with Noisy Negatives) also uses the unlabeled documents as negative documents.

ISVM is the Ideal SVM trained from completely labeled documents (P, with true N which is manually classified from U). ISVM shows the ideal performance when the unlabeled documents are labeled.

We used LIBSVM version 2.36[4] for SVM implementation of SVMC, SMC, SVM_NN, OSVM, and ISVM. We used linear kernel for our experiments because it is simple and powerful enough for text classification problems [3, 5]. For S-EM and NB_NN, we used the recent implementation released by the author of S-EM.[5]

Note that for a conventional TC_WON system, one cannot perform a validation process to optimize the parameters because no negative data is available. For SVMC, SMC, SVM_NN, and ISVM, we used a theoretically motivated fixed parameter ($\nu = 0.01$). We also used the default parameter for S-EM and NB_NN. For OSVM, Tax proposed a sophisticated method to optimize the parameters without negative data [13]. However, their method is infeasibly inefficient especially in high dimensional spaces. We linearly searched for the best parameter for OSVM based on the testing data set in order to simulate the results after the optimization. Note that the true performance of OSVM in practice would be poorer than those reported in our experiments since we unfairly performed the optimization only for OSVM.

For each experiment, we divided the full positive set into two disjoint subsets p_1 and p_2. p_1 is to generate a set of positive examples P, while p_2 is to provide positive examples that can be added to U as noise. For instance, the ModApte split of the Reuters divides the entire set into two subsets – 9603 training and 3299 test documents. We set p_1 and p_2 to the positive data from the training and testing set respectively. For the WebKB data, we set p_1 to a half of the full positive set and p_2 the other half. Note that ISVM trains from P, with true N manually classified from U, which implies that the same negative data are used for both the training and testing set in ISVM. We vary the amount of positive examples (controlled by α) as well as the amount of noise (i.e., positive examples) in the unlabeled data (controlled by β). Specifically, the positive example set P is composed of α fraction of p_1, and the unlabeled data U is composed of β fraction of p_2 and all the negative data. For example, if $\alpha = 1.0$ and $\beta = 0.5$, then 100% of p_1 are used as positive examples, and 50% of p_2 are added to U as noise. Performance is measured on U; we are essentially testing each method's ability of identifying the positive examples in U. We believe that this corresponds to how a method would actually be used for TC-WON in real applications.

4.4 Results and Discussion

4.4.1 Overall results

Table 1 shows the performance results in realistic situations ($\alpha = \beta = 1.0$) where:

- for the WebKB, P is a randomly sampled half of the full positive set, and U is the other half, with a randomly sampled half of the full negative set.
- for the Reuters, P is all the positive data in the training set, and U is all the negative data in the training set, with all data in the testing set.

We observe the following from Table 1.

- SMC and SVMC have the highest performance among all methods trained from positive and unlabeled documents.
- SMC and SVMC have very similar performance, but SVMC always trained much faster than SMC.[6] The difference of the training time between them could vary depending on the number of iterations they undergo.
- The optimized OSVM still performs worse than SMC or SVMC due to the "incomplete" SVs in high dimensional spaces.
- SVM_NN gives very low F_1 scores due to the low recall. The small number of false positives in SVM_NN are likely to affect the support vectors (SVs) of the boundary, which hurts the recall performance.
- S-EM shows the same performance as NB_NN. As noted in [7], S-EM outperforms NB_NN only when the positive documents form a significant fraction of the entire documents. (i.e., when $|P_U|$ is significantly large compared to $|U|$.)

[3] We used Rainbow (http://www-2.cs.cmu.edu/∼mccallum/bow/rainbow/) for text processing.

[4] http://www.csie.ntu.edu.tw/∼cjlin/libsvm

[5] http://www.cs.uic.edu/∼liub/S-EM/readme.html

[6] We ran our experiments on a linux machine of Pentium III 800MHz with 384 MB memory.

Data set	Class	$\|P\|$	$\|U\|$	$\|P_U\|$	F_1 (upper) and accuracy (below)							t-time (sec.)	
					SMC	SVMC	OSVM	SVM_NN	S-EM	NB_NN	ISVM	SMC	SVMC
Reuters	earn	2877	10025	1087	0.9653 0.9923	0.9632 0.9918	0.8267 0.9578	0.0092 0.9645	0.8327 0.9475	0.8327 0.9475	0.9893 0.9977	963.87	119.62
	grain	432	12470	149	0.8239 0.9960	0.8211 0.9959	0.6962 0.9929	0.1250 0.9888	0.4307 0.9720	0.4307 0.9720	0.8905 0.9976	178.30	40.12
	wheat	212	12690	71	0.7737 0.9976	0.7571 0.9973	0.6621 0.9961	0.1519 0.9947	0.2190 0.9623	0.2190 0.9623	0.8413 0.9984	87.75	22.45
	corn	181	12721	56	0.7692 0.9979	0.7521 0.9977	0.4923 0.9948	0.2162 0.9961	0.1571 0.9595	0.1571 0.9595	0.8119 0.9985	78.02	22.19
	crude	389	12513	189	0.7308 0.9922	0.7111 0.9917	0.6685 0.9904	0.0309 0.9850	0.6119 0.9834	0.6119 0.9834	0.8724 0.9966	319.17	54.06
Webkb	course	482	4179	448	0.8274 0.9638	0.8307 0.9640	0.2028 0.3659	0.0880 0.8930	0.5724 0.8411	0.5724 0.8411	0.8969 0.9782	636.70	143.29
	faculty	532	4209	592	0.8468 0.9630	0.8385 0.9611	0.2621 0.5011	0.0168 0.8605	0.5801 0.8532	0.5801 0.8532	0.9032 0.9768	749.63	217.85
	student	816	4154	825	0.8634 0.9485	0.8566 0.9446	0.3384 0.3252	0.0000 0.8014	0.5678 0.7279	0.5678 0.7279	0.9240 0.9727	1181.59	296.50

Table 1: Performance results ($\alpha = \beta = 1.0$). $|P_U|$: # of positives in U; t-time: training time

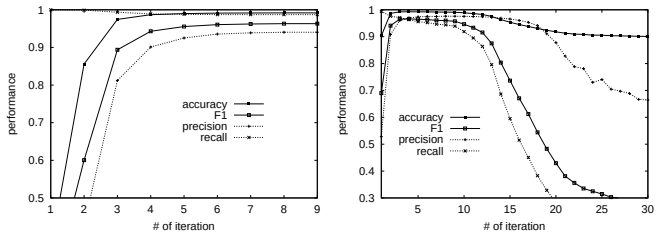
Class	settings	$\|P\|$	$\|U\|$	$\|P_U\|$	F_1 (upper) and accuracy (below)						
					SMC	SVMC	OSVM	SVM_NN	S-EM	NB_NN	ISVM
"earn" in Reuters	$\alpha = 1.0, \beta = 0.7$	2877	9513	575	0.9513 0.9923	0.9500 0.9921	0.7768 0.9595	0.0109 0.9250	0.8120 0.9709	0.8120 0.9709	0.9931 0.9990
	$\alpha = 1.0, \beta = 0.5$	2877	9667	729	0.9427 0.9927	0.9381 0.9921	0.7432 0.9605	0.0069 0.9398	0.8033 0.9747	0.8033 0.9747	0.9877 0.9985
	$\alpha = 1.0, \beta = 0.3$	2877	9289	351	0.9017 0.9919	0.9065 0.9924	0.6426 0.9656	0.0113 0.9624	0.7491 0.9779	0.7491 0.9779	0.9914 0.9994
	$\alpha = 0.7, \beta = 1.0$	1996	10025	1087	0.9646 0.9922	0.9625 0.9917	0.8412 0.9626	0.0073 0.8920	0.8146 0.9621	0.8146 0.9621	0.9855 0.9969
	$\alpha = 0.5, \beta = 1.0$	1420	10025	1087	0.9639 0.9921	0.9621 0.9917	0.8468 0.9674	0.0073 0.8920	0.7900 0.9583	0.7900 0.9583	0.9822 0.9962
	$\alpha = 0.3, \beta = 1.0$	889	10025	1087	0.3667 0.9149	0.2836 0.9012	0.8763 0.9736	0.0037 0.8918	0.7664 0.9559	0.7664 0.9559	0.9774 0.9768
"student" in Webkb	$\alpha = 1.0, \beta = 0.7$	846	3891	552	0.8411 0.9566	0.8418 0.9530	0.2681 0.3418	0.0000 0.8581	0.4843 0.7176	0.4843 0.7176	0.9299 0.9812
	$\alpha = 1.0, \beta = 0.5$	846	3753	414	0.8231 0.9592	0.8161 0.9560	0.2121 0.3208	0.0096 0.8900	0.4186 0.7128	0.4186 0.7128	0.9215 0.9837
	$\alpha = 1.0, \beta = 0.3$	846	3563	224	0.7524 0.9638	0.7273 0.9579	0.1253 0.6290	0.0000 0.9396	0.2855 0.7023	0.2855 0.7023	0.9224 0.9907
	$\alpha = 0.7, \beta = 1.0$	602	4134	795	0.7932 0.9289	0.7914 0.9269	0.3363 0.3449	0.0000 0.8077	0.5916 0.7579	0.5916 0.7579	0.8947 0.9632
	$\alpha = 0.5, \beta = 1.0$	414	4134	795	0.5942 0.8844	0.6889 0.9028	0.3358 0.3370	0.0000 0.8077	0.6093 0.7836	0.6093 0.7836	0.8326 0.9448
	$\alpha = 0.3, \beta = 1.0$	276	4134	795	0.0840 0.8153	0.3810 0.8491	0.3353 0.3220	0.0000 0.8077	0.5767 0.7512	0.5767 0.7512	0.7411 0.9209

Table 2: Performance results for different settings. $|P_U|$: # of positives in U

4.4.2 Results for different settings

Here we vary α and β to create different environments. Several observations can be made from Table 2:

- SMC and SVMC perform worse than S-EM or NB_NN when α is very low meaning that the positive training data are seriously under-sampled. The reason that the performance of SMC and SVMC is susceptible to a low α is that when the positive training data are too few, the final boundary of SMC and SVMC would trespass the true boundary and result in fitting around the under-sampled data. In this case, the intermediate boundaries of SMC or SVMC generated in the middle of the iterations give better results. (See Figure 5(b).) However, we are not able to optimize the best number of iterations because a TC_WON system is assumed to have no labeled negative data available for training or optimization. Optimizing the number of iterations of SVMC with positive and unlabeled data would be an interesting further work to improve the performance of SVMC in case of a low α.
- β does not seem to influence the performance of SMC and SVMC significantly, as long as the positive data is not under-sampled because the false negatives in U will be excluded in the training of the final boundary.
- OSVM shows stable performance less affected by α or β. However, the stable performance is achieved by the unfairly performed optimization, which is impossible to do in practice in high dimensional spaces.
- As expected, SVM_NN suffers from very low recall performance, and almost always gives a nearly zero F_1 score.
- S-EM still shows the same performance as NB_NN, which implies that S-EM might not be useful for common positive classes as in our experiments. ([7] gives an example of "the class of movies we may enjoy watching" as a dominant positive class where S-EM might outperform NB_NN.)



(a) $\alpha = \beta = 1.0$ (b) $\alpha = 0.3$ and $\beta = 1.0$

Figure 5: SVMC performance convergence on the "earn" class

4.4.3 Performance convergence of SVMC

Figure 5(a) shows the SVMC performance convergence on the "earn" class when the positive documents are fairly large. After the first iteration, the recall was very high, but the precision was very low because the initial class boundary covers a wide area including many false positives in the feature space. As it iterates, the precision and the overall performance increases rapidly with a little decrease of the recall. Our experiments on all other classes in Table 1 also show similar shapes of the convergence graph.

Figure 5(b) shows the SVMC performance convergence when the positive documents are seriously under-sampled. The convergence happens after 30 iterations, but the highest performance is actually achieved at 5th iteration. This suggests that when a sufficiently large number of positive documents are available, SVMC is more robust; otherwise, it may "over-iterate" and result in poor performance. How to improve SVMC's performance with very few positive examples is clearly an important direction for future research.

5. CONCLUSIONS AND FURTHER WORK

We study text classification from positive and unlabeled examples, which is a fundamental task related with several different information problems, such as text categorization, information filtering, and relevance feedback. We explore the MC and SVMC algorithms which compute an accurate classification boundary of the positive documents without relying on labeled negative documents by iteratively applying a margin maximization classifier, such as SVM, to progressively improved negative documents. We evaluate MC and SVMC using the commonly used data sets Reuters-21578 and WebKB. Experiment results show that with a reasonable amount of positive documents, the MC algorithm gives the best performance among all the existing TC_WON methods that we compared with. When the positive training data is seriously under-sampled, the boundary of MC or SVMC tends to be over-conservative and much tighter than the true optimal boundary; in such a case, the intermediate boundaries of them before converging give much better results. It would be interesting to investigate an optimization technique to uncover the best number of iterations of the MC and SVMC algorithms given positive and unlabeled documents.

6. REFERENCES

[1] K. M. A. Chai, H. T. Ng, and H. L. Chieu. Bayesian online classifiers for text classfication and filtering. In *Proc. 25th ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'02)*, pages 97–104, Tampere, Finland, 2002.

[2] C.-C. Chang and C.-J. Lin. Training nu-support vector classifiers: theory and algorithms. *Neural Computation*, 13:2119–2147, 2001.

[3] T. Joachims. Text categorization with support vector machines. In *Proc. 10th European Conference on Machine Learning (ECML'98)*, pages 137–142, Chemnitz, Germany, 1998.

[4] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. 16th Int. Conf. Machine Learning (ICML'00)*, pages 200–209, Bled, Slovenia, 1999.

[5] T. Joachims. A statistical learning model of text classification with support vector machines. In *Proceedings of SIGIR-01, 24th ACM Int. Conf. on Research and Development in Information Retrieval*, pages 128–136, New Orleans, US, 2001.

[6] D. D. Lewis. *Representation and Learning in Information Retrieval*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1992.

[7] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *Proc. 19th Int. Conf. Machine Learning (ICML'02)*, pages 387–394, Sydney, Australia, 2002.

[8] L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[9] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *The Sixteenth Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, 1999.

[10] K. Nigam. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

[11] B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1083–1121, 2000.

[12] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[13] D. M. J. Tax and R. P. W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research*, 2:155–173, 2001.

[14] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[15] Y. Yang. A study on thresholding strategies for text categorization. In *Proc. 24th ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'01)*, pages 137–145, New Orleans, Louisiana, 2001.

[16] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. 22th ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, Berkeley, CA, 1999.

[17] H. Yu. SVMC: Single-class classification with support vector machines. In *Proc. Int. Joint Conf. on Articial Intelligence (IJCAI-03)*, Acapulco, Maxico, 2003.

[18] H. Yu, J. Han, and K. C. Chang. PEBL: Positive-example based learning for Web page classification using SVM. In *Proc. 8th Int. Conf. Knowledge Discovery and Data Mining (KDD'02)*, pages 239–248, Edmonton, Canada, 2002.