
Machine Learning, 61, 49–69, 2005

DOI: 10.1007/s10994-005-1122-7

Single-Class Classification with Mapping Convergence*

HWANJO YU hwanjoyu@cs.uiowa.edu
Department of Computer Science, University of Iowa, Iowa City, IA 52242, USA

Editor: Philip M. Long
Published online: 09 June 2005

Abstract. Single-Class Classification (SCC) seeks to distinguish one class of data from universal set of multiple classes. We call the target class *positive* and the complement set of samples *negative*. In SCC problems, it is assumed that a reasonable sample of the negative data is not available. SCC problems are prevalent in the real world where positive and unlabeled data are widely available but negative data are hard or expensive to acquire. We present an SCC algorithm called *Mapping Convergence* (*MC*) that computes an accurate boundary of the target class from positive and unlabeled data (without labeled negative data). The basic idea of MC is to exploit the natural "gap" between positive and negative data by incrementally labeling negative data from the unlabeled data using the *margin maximization* property of SVM. We also present *Support Vector Mapping Convergence* (*SVMC*) which optimizes the MC algorithm for fast training. Our analyses show that MC and SVMC without labeled negative data significantly outperform other SCC methods. They generate as accurate boundaries as standard SVM with fully labeled data when the positive data is not very under-sampled and there exist gaps between positive and negative classes in the feature space. Our results also show that SVMC trains much faster than MC with very close accuracy.

Keywords: Single-Class Classification, Support Vector Machines

1. Introduction

Single-Class Classification (SCC) seeks to distinguish one class of data from universal set of multiple classes (e.g., distinguishing apples from fruits, identifying waterfall pictures from image databases, or classifying personal homepages from the Web). Throughout the paper, we call the target class *positive* and the complement set of samples *negative*.

In SCC problems, it is assumed that a reasonable sample of the negative data is hard to acquire. Since it is not natural to collect the "non-interesting" objects (i.e., negative data) to train the concept of the "interesting" objects (i.e., positive data), SCC problems are prevalent in real-world applications where positive and unlabeled data are widely available but negative data are hard or expensive to acquire (Yu, Han, & Chang, 2002; Letouzey, Denis, & Gilleron, 2000; DeComite, Denis, & Gilleron, 1999). For example, in text or Web page classification (e.g., personal homepage classification), collecting negative training data

*This paper is based on the paper: *H. Yu "SVMC: Single-Class Classification With Support Vector Machines"*, Proc. 2003 International Joint Conferences on Artificial Intelligence (IJCAI'03). However, this submission is substantially extended and revised in technical contents and contains new experimental results and discussions in comparison with our conference publication.

(e.g., a sample of "non-homepages") is delicate and arduous because manually collected negative data could be easily biased because of a person's unintentional prejudice, which could be detrimental to classification accuracy. For another example, consider the automatic diagnosis of diseases: unlabeled data are easy to collect (all patients in the database), and positive data are also readily available (the patients who have the disease). However, negative data can be expensive to acquire; not all patients in the database can be assumed to be negative if they have not been tested for the disease, and such tests can be expensive. Other applications can be found in pattern recognition, image retrieval, classification for data mining, rare class classification, etc. In this paper, we focus on SCC problem with positive and unlabeled data and without labeled negative data.

1.1. Previous approaches for SCC

Traditional (semi-)supervised learning schemes are not suitable for SCC without labeled negative data because: (1) the portions of positive and negative spaces are seriously unbalanced (i.e., $Pr(P) \ll Pr(\bar{P})$) and hard to estimate, and (2) the absence of negative samples in the labeled data set makes unfair the initial parameters of the model and leads to unfair guesses for the unlabeled data.

Active learning methods try to minimize the labeling labor to construct an accurate classification function by a different approach that involves an interactive process between the learning system and a user (Tong & Koller, 2000).

Valiant in 1984 (Valiant, 1984) pioneered *learning theory from positive examples* based on rule learning. In 1998, F. Denis defined the Probably Approximately Correct (PAC) learning model for positive and unlabeled examples, and showed that k-DNF (Disjunctive Normal Form) is learnable from positive and unlabeled examples (Denis, 1998). After that, attempts to learn using positive and unlabeled data have tried k-DNF or C4.5 (Letouzey, Denis, & Gilleron, 2000; DeComite, Denis, & Gilleron, 1999). Such rule learning methods are simple and efficient for learning nominal features but are tricky to use for problems of continuous features, high dimensions, or sparse instance spaces.

The Positive Example-Based Learning (PEBL) framework was proposed for Web page classification (Yu, Han, & Chang, 2002). Their method is limited to the Web domain with binary features, and its training efficiency is poor as it uses SVM iteratively whose training time is already at least quadratic to the size of training data set. This problem becomes critical when the size of unlabeled data set is large.

Recently, a probabilistic method built upon the EM algorithm for the SCC problem, called *S-EM*, was proposed for the text domain (Liu et al., 2002). The method has several fundamental limitations: the generative model assumption, the attribute independence assumption which results in linear separation, and the requirement of good estimation of prior probabilities. Our method does not require the prior probability of each class, and it can draw nonlinear boundaries using the SVM kernel trick.

The pattern recognition and verification fields have also explored various SCC methods, including neural network models (Frosini, Gori, & Priami, 1996; Gori, Lastrucci, & Soda, 1995) and the SVMs (Manevitz & Yousef, 2001; Bileschi & Heisele, 2003) (with increasing popularity). Some of these techniques tend to be domain specific: For instance, Bileschi &

Heisele (2003) use SVM with only positive examples for face detection—However, it relies on using the face features of other non-target classes as negative examples, and thus is not generally applicable to other domains.

For document classification, Manevitz (Manevitz & Yousef, 2001) compared various SCC methods including neural network method, one-class SVM, nearest neighbor, naive Bayes, and Rocchio, and concluded that One-class SVM and neural network method were superior to all the other methods, and the two are comparable.

OSVM (One-Class SVM), based on the strong mathematical foundation of SVM, distinguishes one class of data from the rest of the feature space given only a positive data set (Scholkopf et al., 2001; Tax and Duin, 2001; Manevitz & Yousef, 2001). OSVM draws a nonlinear boundary of the positive data set in the feature space using two parameters—ν (to control the noise in the training data) and γ (to control the "smoothness" of the boundary). They have the same advantages of SVM, such as efficient handling of high dimensional spaces and systematic nonlinear classification using the kernel trick. However, given no negative examples available, OSVM requires a much larger amount of positive training data to induce an accurate class boundary especially in high dimensional spaces because its support vectors (SVs) of the boundary only comes from the positive data set and thus the relatively small number of SVs can hardly cover the major directions of the boundary in high dimensional spaces. Due to the SVs coming only from positive data, OSVM also tends to overfit and underfit easily. Tax proposed a sophisticated method which uses artificially generated unlabeled data to optimize the OSVM's parameters that "balance" between overfitting and underfitting (Tax and Duin, 2001). However, their optimization method is infeasibly inefficient in high dimensional spaces, and even with the best parameter setting, its performance still lags far behind the SVM with negative data due to the shortage of SVs which makes "incomplete" the boundary description. Figures 1(a) and (b) show the boundaries of SVM trained from positives and negatives and OSVM trained from only positives on a synthetic data set in a two-dimensional space. (We used LIBSVM version 2.33[1] for SVM implementation. The RBF kernel parameter is optimized on training data.) In this low-dimensional space with "enough" data, the obstensibly "smooth" boundary of OSVM is not the result of the good generalization but instead is from the "incomplete" SVs due to not using the negatives for SVs, which will become much worse in high-dimensional spaces where more SVs around the boundary are needed to cover major directions in the high-dimensional spaces. When we increase the number of SVs in OSVM, it overfits rather than being more accurate as shown in Figures 1(c) and (d).

However, such OSVM boundary might be the best achievable one when only positive data are available. In this paper, under the assumption that unlabeled data are abundant and readily available, we present a systematic method that additionally uses the unlabeled data for support vectors to get more concrete boundary descriptions.

1.2. Contributions and paper layout

We present an SCC algorithm called *Mapping Convergence* (*MC*) that computes an accurate boundary of the target class from positive and unlabeled data (without labeled negative data) (Yu, 2003). The basic idea of MC is to exploit the natural "gap" between positive and negative

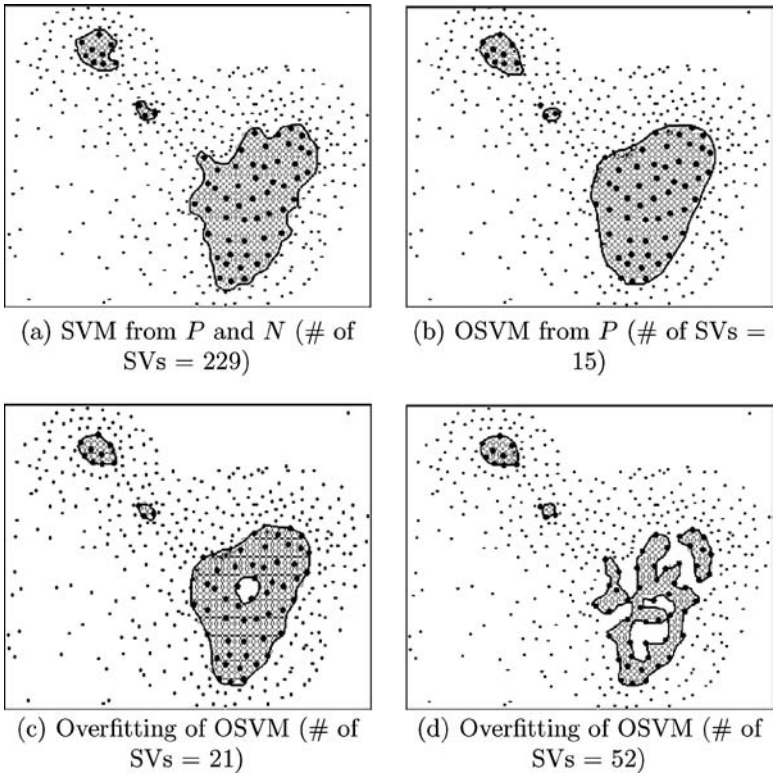


Figure 1. Boundaries of SVM and OSVM on a synthetic data set. *big dots: positive data*, *small dots: negative data.*

data by incrementally labeling negative data from the unlabeled data using the *margin maximization* property of SVM. We also present *Support Vector Mapping Convergence* (*SVMC*) which optimizes the MC algorithm for fast training. Our analyses show that MC and SVMC without labeled negative data significantly outperform other SCC methods. They generate as accurate boundaries as standard SVM with fully labeled data when the positive data is not very under-sampled and there exist gaps between positive and negative classes in the feature space. Our results also show that SVMC trains much faster than MC with very close accuracy.

In Section 2, we first discuss the optimal SCC boundary, which motivates our SCC method, *MC*, and present the MC algorithm which generates the boundary close to the optimum under certain conditions. We motivate and describe the SVMC algorithm in Section 3. In Section 4, we empirically verify our analysis of SVMC by extensive experiments on various domains of real data sets such as text classification, letter recognition, diagnosis of breast cancer, which shows the outstanding performance of SVMC in a wide spectrum of SCC problems (with linear or nonlinear separation, and low or high dimensions).

2. Mapping Convergence (MC) algorithm

In this section, we present the basic idea of the Mapping-Convergence (MC) algorithm, which is the basis of the SVMC algorithm. 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 an iid (independently and identically distributed) sample of the universal set.
- x is a data instance such that $x \in \mathcal{U}$.
- $\mathcal{P}$ is a space 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.

2.1. Motivation

In machine learning theory, the *optimal* class boundary function (or hypothesis) $h(x)$, given a limited number of training data $\{(x, l)\}$ (l is the label of x), is the one that gives the best *generalization* performance which is the performance on "unseen" examples rather than on the training data. The performance on the training data is not regarded as a good evaluation measure for a hypothesis because the hypothesis may end up *overfitting* when it tries too hard to fit the training data. When a problem is easy (to classify) and the boundary function is more complicated than it needs to be, the boundary is likely overfitting. When a problem is hard and the classifier is not powerful enough, the boundary is likely underfitting. SVM is an excellent example of supervised learning that tries to maximize the generalization by maximizing the *margin* and also supports nonlinear separation using the kernel trick, by which SVM tries to avoid overfitting and underfitting (Vapnik, 1998; Christianini & Shawe-Taylor, 2000; Burges, 1998).

To find the optimal SCC classifier without labeled negative data, we also needs to maximize the generalization performance so that we do not overfit nor underfit the training data. The optimal SCC boundary is much harder to find than one with complete labels for all the training data. To illustrate an example of desirable SCC boundary when no labeled negative data is available, consider the synthetic data set (Figure 2) which simulates a real situation where (1) $\mathcal{U}$ is the universal set consisting of multiple groups of data, (2) the positive class $\mathcal{P}$ is one of them (say the data group in the center), and (3) the positive data set P (represented by the big dots) is a sample from $\mathcal{P}$. OSVM draws a very conservative tight boundary around P as shown in Figure 2(a), clearly overfitting the data due to its inability of using any knowledge about the distribution of U. Intuitively, the desirable boundary should be located between positive class $\mathcal{P}$ and the unlabeled data U outside $\mathcal{P}$. The MC algorithm exploits U systematically to obtain a reasonable set of negative data, which allows us to reach such a desirable boundary shown in Figure 2(b).

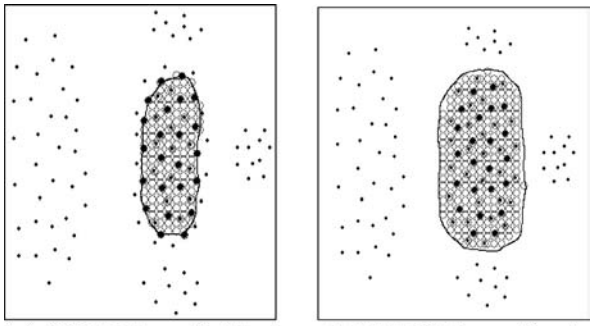


Figure 2. Synthetic data set simulating a real situation. *P: big dots*, *U: all dots (big and small dots).*

2.2. *Strong negative*

Here we introduce the notion of "strong negative", which is key to the 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

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

Definition 1 (Strong negative). 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 a stronger negative (or 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.

2.3. *MC algorithm*

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), to draw an initial approximation of "strong negatives"—the negative data located far away from the boundary of the positive class in $\mathcal{U}$ (Steps 1 and 2 in Figure 3). Based on this initial approximation, the convergence stage runs iteratively using a second base

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 3. MC algorithm.

classifier Ψ_2 (e.g., SVM), to maximize the margin in order to make a progressively better approximation of negative data (steps 3 through 5 in Figure 3). As a result, the class boundary eventually converges to the boundary around the positive data set in the feature space. The MC algorithm is described in Figure 3.

To illustrate the MC process, consider an example of data distribution in a one-dimensional feature space in Figure 4. The unlabeled data U is composed of eight data clusters. 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) in Figure 4, where b_d maximizes the margin—the gap between the two points g_n and g_f—between positive and negative clusters. Unfortunately, under the common scenarios of SCC, the only labeled data are the dark center within the positive cluster, which is a subset of the positive class, and the rest are unlabeled.

Figure 5(a) illustrates the MC process given the labeled positive data P (i.e., the dark center) with the unlabeled U. First, algorithm Ψ_1 identifies *strong negatives*; in Figure 5(a), let *strong negatives* be the data to the left and right sides of b_1 and b'_1 respectively, i.e., the dark area at the edges. They 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. For convenience of explanation, consider only the left side of the positive cluster in Figure 4(a). After we identified the strong

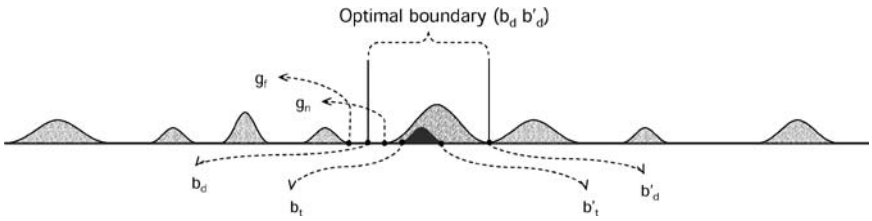
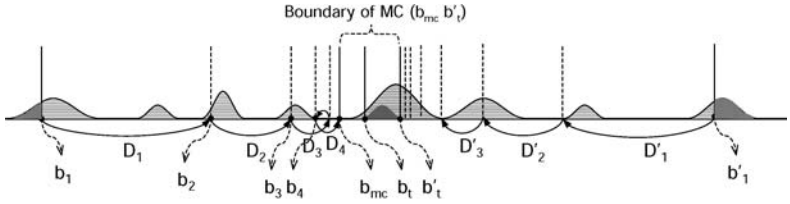


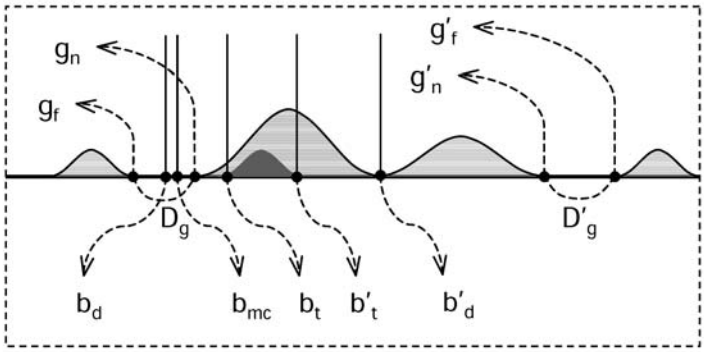
Figure 4. Example of data distribution in one dimensional feature space. The fifth data cluster from left is positive and the rest are negative.

negatives, at the first iteration of Step 4 in Figure 3, algorithm Ψ_2 computes, from P and the strong negatives N, an classification boundary h_1 (i.e., b_2 in Figure 5(a)). Then, the data classified as negative by h_1 (i.e., the data within D_1 in Figure 5(a)) is merged into N. With the revised N and the original P, Ψ_2 computes another boundary h_2 (i.e., b_3 in the figure) and negatively classified data (i.e., the data within D_2) is merged again into N. As the iteration goes, N progressively includes the unlabeled data as negative data, and the boundary h_i approaches to the optimal boundary.

Assume for now that Ψ_2 be a *hard*-margin SVM. Let D_i be the margin for the negatively classified data after ith iteration. Since Ψ_2 at each iteration computes a boundary that maximizes the margin, D_{i+1} will have half the margin of D_i in Figure 5(a) except one case:



(a) MC process.



(b) Zoomed-in center.

Figure 5. Example of the MC algorithm in the one dimensional feature space.

Suppose b_i is the starting point of D_i that is merged into N at ith iteration, and let b_t be the boundary of the positive data P, as Figure 5(a) illustrates. If there exists *no data around* the boundary b_{i+1} *after* ith iteration, b_{i+1} will retract to a point where there *exists* 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 5(a), b_4 retracts a little from the boundary after the 3rd iteration, as there exists no data around the boundary after the 3rd iteration.

If there exists *no data at all* within D_i, the convergence will stop because nothing will be added to N and thus more iterations after that will not affect the boundary. For instance, in Figure 5(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 included into N from that point. As we see from Step 4.5 in Figure 3, the MC algorithm will stop the convergence when nothing is included into N. The final boundary of MC *to the right side* in Figure 5 tightly fits around P (i.e., b'_t) because there is no gap between the positive and negative clusters and thus the convergence will not stop until MC includes 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., The MC boundary b_{mc} is close to the optimal boundary b_d in Figure 5(b).) The optimal boundary b_d equally divides the margin between the points g_f and g_n in Figure 5(b), i.e., the margin between the positive and negative data clusters, while b_{mc} does between two points g_f and b_t, as only the data in the dark center is labeled. However, if there is no wide gap between those (e.g., the right side of Figure 5(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 5(b)).

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 5(a)). We denote $|x - y|$ for the margin between two points x and y. Let D_g be a 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 illustrated in Figure 5(b). Then we have the following lemma.

Lemma 1. *Suppose g_f is the starting point at ith iteration. If MC stop converging at ith iteration, there must exists a gap D_g such that $|g_f - g_n| > |g_n - b_t|$.*

Proof:

If MC stops the convergence at ith iteration, there must exists a gap at least from g_f to the half point to b_t because Ψ_2 equally divides the margin between g_f and b_t, i.e., $|g_f - g_n| > \frac{|g_f - b_t|}{2} = \frac{|g_f - g_n| + |g_n - b_t|}{2}$. Thus $|g_f - g_n| > |g_n - b_t|$. □

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

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 $\frac{|g_f - b_t|}{2}$ from 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. □

Note that our algorithm analyses so far assume that data is noiseless and Ψ_2 is a hard-margin SVM. In practice, we use a soft-margin SVM for Ψ_2 to deal with noise in the training data. With soft-margin SVMs, D_{i+1} may not exactly have the half margin of D_i. However, a small soft-margin parameter (e.g., $\nu = 0.01$ *or* 0.1 in ν-SVM) usually performs well in SCC, as our experiments in Section 4 also verify, because in SCC, P is likely to be a carefully collected subset of positive data and thus likely to be noise-free.

Theorem 1 can be interpreted as carrying the following messages.

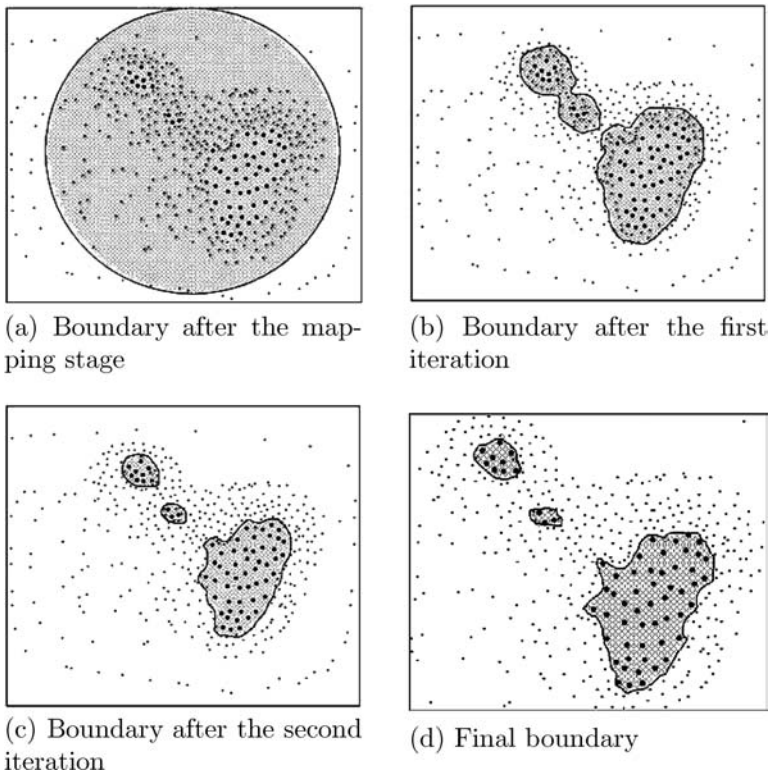
1. MC is not likely to locate the boundary exorbitantly far from the boundary of P unless U is extremely sparse, because a larger gap ($|g_f - g_n|$) is needed to stop the convergence as g_n (i.e., the nearer ending point of the gap) is becoming farther from b_t (or $|g_n - b_t|$ increases).
2. The more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

The above observations imply that MC is likely to generate the boundary close to the optimum when P is not seriously under-sampled and wide gaps exist between P and N in the feature space. Data sets in practice are likely to have natural gaps between different classes than within a class in the feature space. From the first observation, gaps far from P in the feature space are less likely to influence MC. By exploiting such gaps, MC significantly outperforms other SCC methods when P is not too much under-sampled. Our experiments on common data sets in Section 4 show consistent results.

What are the conditions that the component algorithms Ψ_1 and Ψ_2 must meet, in order for MC to perform well?

1. Ψ_1 *excludes false negatives while including as many strong negatives.* First, if Ψ_1 includes false negatives, they are likely to stay as false negatives during the iterations which will deteriorate the boundary accuracy. Second, an insufficient amount of strong negatives might not cover all the directions in the feature space, which could generate "incomplete" set of negative SVs especially in a high-dimensional feature space. Thus, more negatives Ψ_1 includes, MC will generate the more stable boundary convergence.

   We can use any reasonable classifier for Ψ_1, such as Rocchio or OSVM, and adjust the threshold so that it makes near 100% recall with a minimal sacrifice of precision. We use OSVM for Ψ_1 in our research, and adjust the bias b to achieve a high recall. Figure 6 shows an example of the boundary after each iteration of SVMC: Ψ_1 identifies strong negatives by covering a wide area around the positives (Figure 6(a)). Although the precision quality of the mapping is poor, the boundary at each iteration converges



(a) Boundary after the mapping stage

(b) Boundary after the first iteration

(c) Boundary after the second iteration

(d) Final boundary

Figure 6. Intermediate results of SVMC.

(Figures 6(b) and (c)), and the final boundary is close to the ideal boundary drawn by SVM on P and *true* N. (Compare Figures 1(a) and 6(d)). We used the same kernel parameter for both SVMC (Figure 6) and the ideal SVM (Figure 1(a)), as the boundaries of both are likely to have similar number of SVs. (We will discuss tuning parameters and also compare the number of SVs generated by different methods in Section 4.) Our experiments in Section 4 also show that the final boundary becomes accurate although the initial boundary of the mapping stage is relatively rough by the "loose" setting of the threshold for Ψ_1.

2. Ψ_2 *maximizes the margin.* Without the margin maximization of Ψ_2, the boundary convergence could stop in an arbitrary gap in the feature space. In other words, the convergence behavior would be inconsistent to the type of data distribution if Ψ_2 does not provide the margin maximization property, because Theorem 1 becomes invalid without the property.

   SVM and Boosting are currently the most popular supervised learning algorithms that maximize margin. We use SVM for Ψ_2 in 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 (Scholkopf et al., 2000; Chang and Lin, 2001)).

3. Support Vector Mapping Convergence (SVMC)

In this section, we present *Support Vector Mapping Convergence* (*SVMC*) which optimizes the MC algorithm for fast training.

3.1. Motivation

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 Chang and Lin (2001)). 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. Here we introduce a trick in the SVMC algorithm, to significantly reduce the training time while keeping the accuracy from degrading.

3.2. SVMC algorithm

The basic idea of SVMC is to remove the data, from the training set at each iteration, that are *unlikely* to be the SVs. To illustrate, consider the point of starting the third iteration (when $i = 2$) in MC (Step 4.1 in Figure 3) illustrated in Figure 7. 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 is unlikely to be the SVs. The negative SVs of h_2 fairly represent the data within $\hat{N}_0$ and $\hat{N}_1$. Thus, 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.

The SVMC algorithm is described in Figure 8. Surprisingly, adding only Step 4.5 to the original MC algorithm completes SVMC. Our experiments in Section 4 show that SVMC trains much faster than MC and its accuracy is almost as high as that of MC.

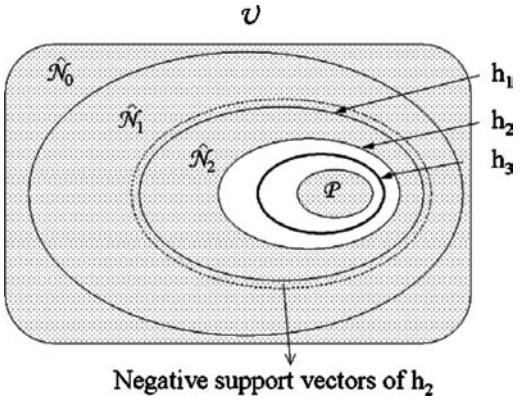


Figure 7. Minimally required negative data for h_3.

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

Ψ_1: Rocchio or OSVM
Ψ_2: 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. Reset N with negative SVs of h_{i+1}
   4.6. Repeat until $\hat{N}_i = \emptyset$
5. return h_i

Figure 8. SVMC algorithm.

Note that, while this heuristics significantly reduces the training time (See Table 1 for comparing the training time.), it could marginally deteriorate the quality of the final boundary, because the negative SVs of h_i cannot entirely represent the negative data accumulated in the previous iterations. Depending on the data distribution and the boundary, any negative data from previous iterations could be SVs for h_i. From Table 1, we see the SVMC's final boundary is slightly less accurate than that of MC for most datasets.

Table 1. Performance results ($\alpha = \beta = 1.0$). *t*-time: training time.

	F_1 (1st row), accuracy (2nd row), No. of SVs (3rd row)							*t*-time (sec.)	
Class	MC	SVMC	OSVM	SVM_NN	S-EM	NB_NN	ISVM	MC	SVMC
Earn	0.9653	0.9632	0.8267	0.0092	0.8327	0.8327	0.9893	963.87	119.62
	0.9923	0.9918	0.9578	0.8921	0.9475	0.9475	0.9977		
	(1209)	(1058)	(374)	(1948)	–	–	(1264)		
Grain	0.8239	0.8211	0.6962	0.1250	0.4307	0.4307	0.8905	178.30	40.12
	0.9960	0.9959	0.9929	0.9888	0.9720	0.9720	0.9976		
	(636)	(493)	(143)	(788)	–	–	(642)		
Wheat	0.7737	0.7571	0.6621	0.1519	0.2190	0.2190	0.8413	87.75	22.45
	0.9976	0.9973	0.9961	0.9947	0.9623	0.9623	0.9984		
	(349)	(259)	(63)	(441)	–	–	(370)		
Corn	0.7692	0.7521	0.4923	0.2162	0.1571	0.1571	0.8119	78.02	22.19
	0.9979	0.9977	0.9948	0.9961	0.9595	0.9595	0.9985		
	(400)	(313)	(68)	(463)	–	–	(397)		
Crude	0.7308	0.7111	0.6685	0.0309	0.6119	0.6119	0.8724	319.17	54.06
	0.9922	0.9917	0.9904	0.9850	0.9834	0.9834	0.9966		
	(689)	(527)	(121)	(851)	–	–	(656)		
Letter 'A'	0.9840	0.9840	0.8457	0.0811	0.4757	0.4757	0.9929	171.77	45.37
	0.9991	0.9991	0.9922	0.9726	0.9467	0.9467	0.9996		
	(1385)	(1048)	(150)	(891)	–	–	(1407)		
Letter 'B'	0.9046	0.9204	0.7207	0.0834	0.1725	0.1725	0.9651	75.61	14.34
	0.9950	0.9959	0.9869	0.9758	0.8489	0.8489	0.9983		
	(619)	(529)	(193)	(905)	–	–	(618)		
Letter 'C'	0.9641	0.9641	0.7354	0.0758	0.1229	0.1229	0.9860	155.70	29.93
	0.9982	0.9982	0.9882	0.9755	0.7293	0.7293	0.9923		
	(1202)	(733)	(207)	(908)	–	–	(1205)		
Letter 'D'	0.9300	0.9300	0.6921	0.0902	0.1610	0.1610	0.9820	80.47	16.06
	0.9963	0.9963	0.9860	0.9752	0.8881	0.8881	0.9991		
	(651)	(554)	(207)	(930)	–	–	(715)		
Letter 'E'	0.9419	0.9396	0.7112	0.1333	0.1861	0.1861	0.9798	98.13	21.57
	0.9970	0.9969	0.9878	0.9764	0.8617	0.8617	0.9990		
	(698)	(567)	(186)	(885)	–	–	(709)		
b-cancer	0.9470	0.8444	0.6315	0.1872	0.4741	0.4741	0.9521	0.131	0.025
	0.9841	0.9658	0.8332	0.2664	0.6659	0.6659	0.9858		
	(56)	(23)	(12)	(116)	–	–	(55)		

4. Experimental evaluation

In this section, we provide the empirical evaluations on MC and SVMC by performing extensive experiments on various domains of real data sets—text classification, letter recognition, and diagnosis of breast cancer—which show the outstanding performance of SVMC in a wide spectrum of SCC problems (with linear or nonlinear separation, and low or high dimensions). Our evaluations show that MC and SVMC without labeled negative data significantly outperform other SCC methods and in most cases generate as accurate boundaries as standard SVM with fully labeled data, except when the positive data set is seriously under-sampled. Our results also show that SVMC trains much faster than MC with very close accuracy.

4.1. Performance measures

To evaluate a SCC method, we use the F_1 measure, which is a commonly used performance measure for SCC and also used (Liu et al., 2002)—one of the most recent works on SCC for text classification. This measure combines precision and recall in the following way:

$$\text{precision} = \frac{\text{No. of correct positive predictions}}{\text{No. of positive preditions}}$$

$$\text{recall} = \frac{\text{No. of correct positive predictions}}{\text{No. of positive examples}}$$

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

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

4.2. Experiment methodology

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

- OSVM is described in Section 1.1.
- SVM_NN is standard SVM trained using positive data, with unlabeled data as a subsititute for negative data. This is not unreasonable, as the unlabeled data can be thought of as a good approximation of negative data. However, the small number of false negatives are likely to affect the support vectors (SVs) of the boundary, which hurts the recall performance, as shown in our experiments.
- S-EM is described in Section 1.1.
- NB_NN (Naive Bayes with Noisy Negatives) also uses the unlabeled data as negative data.
- ISVM is the Ideal SVM trained from completely labeled data (P, with true N which is manually classified from U). ISVM shows the ideal performance when the unlabeled data are labeled.

We used LIBSVM version 2.36[2] for SVM implementation of SVMC, MC, SVM_NN, OSVM, and ISVM. For the Reuters and breast cancer data sets, we used linear kernels for which are simple and accurate enough. (Text classification tasks are generally linearly separable (Joachims, 2001)). For the letter recognition data set, we use Gaussian kernels which usually perform the best for pattern recognition problems.

For S-EM and NB_NN, we used the recent implementation released by the author of S-EM.[3]

Note that for a conventional SCC method, one cannot perform a validation process to optimize the parameters because no negative data is available. For SVMC, MC, SVM_NN,

and ISVM, we used a theoretically motivated fixed parameter (e.g., $\nu = 0.01$ *or* 0.1). As discussed at the end of Section 2.3, 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 since P is not likely noisy. (We used ν-SVM for the semantically meaningful parameter (Scholkopf et al., 2000; Chang and Lin, 2001)). For the Gaussian kernel parameter γ, we used the default value $\gamma = 1/m$ where m is the number of dimensions. The default γ performed very well on the letter recognition data set. However, kernel parameters such as γ usually affects the performance very much and thus need a careful optimization. Optimizing the kernel parameters for MC with only positive and unlabeled data is an important further work.

We also used the default parameter for S-EM, as it is impossible to optimize the parameter without negative data (Refer to Liu et al. (2002) for details.).

For OSVM, Tax proposed a sophisticated method to optimize the parameters without negative data (Tax and Duin, 2001). However, their method is infeasibly inefficient especially in high dimensional spaces. As OSVM performs very poorly with default parameters, 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. Even with the unfairly optimized parameter, OSVM performs poorer in our experiments than MC or SVMC with the intuitively set parameters.

4.3. Data sets

We used three data sets from three different domains: Reuters-21578.[4] for text classification, and for pattern recognition and bioinformatics, the letter recognition and breast cancer data sets from the UCI machine learning repository[5]

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.[6] 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. We choose five frequently occurring categories for our experiments—*earn*, *grain*, *wheat*, *corn*, *crude*. Experiment on each category is performed independently.

The letter recognition data set includes 20000 samples of 26 capital letters, with each letter having about 800 samples. Each sample is represented by 16 numerical features of statistical moments and edge counts. We choose the first five letters for our experiments—*A*, *B*, *C*, *D*, *E*.

The breast cancer data set has 569 samples—*357 benign and 212 malignant*. Each sample is represented by 30 real-valued features that are the descriptions of each cell. We classify the malignant cells as positive class.

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 unlabeled data. 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 letter recognition

and breast cancer data sets, we set p_1 to two third of the full positive set and p_2 to the other one third.

We vary the amount of positive examples (controlled by α) as well as the amount of unlabeled positive examples (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. Performance is measured on U; we are essentially testing each method's ability of identifying the positive examples in U. Our experiment setup is very similar to that in Liu et al. (2002), except our setup is more realistic: In Liu et al. (2002), U is composed of $\beta\%$ of positives (e.g., class 'A') and samples from another class (e.g., class 'B'). Our U is $\beta\%$ of positives and the remainder is from all other classes. 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.

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 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.
- for the letter recognition data set, P is a randomly sampled two third of the full positive set, and U is the other one third, with a randomly sampled half of the full negative set. The results are averaged from five runs.
- for the breast cancer data set, P is a randomly sampled two third of the full positive set, and U is all others. The results are averaged from five runs.

We have the following observations from Table 1.

- MC and SVMC have the highest performance among all the methods trained from positive and unlabeled data. Although MC and SVMC show very similar performance, MC mostly performed a little better than SVMC as discussed in Section 3.2. The number of SVs of MC is also very close to that of ISVM while that of SVMC is a little smaller.
- SVMC always trained much faster than MC.[7] 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 MC or SVMC due to the inability of using the unlabeled data for SVs. The number of SVs of OSVM is much smaller than others since they use only positives for SVs.
- 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 SVs of the boundary, which hurts the recall performance badly.

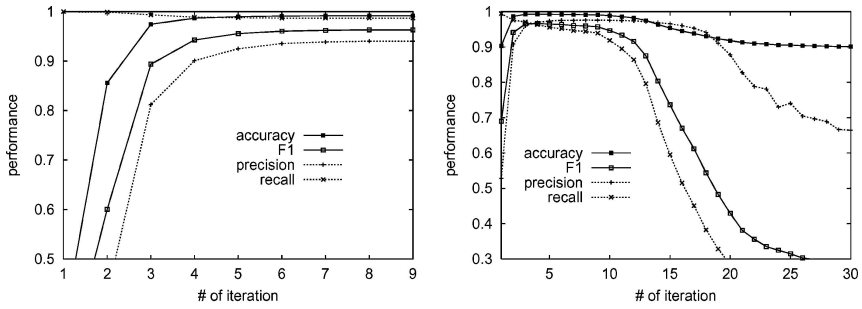
– S-EM shows the same performance as NB_NN. As noted in Liu et al. (2002), S-EM outperforms NB_NN only when the positive data form a significant fraction of the universal set (i.e., when $|P_U|$ is significantly large compared to $|U|$).

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:

– The performance of MC and SVMC drops abruptly when α is set very low. (In Table 2, the F_1 values are dropped significantly at $\alpha = 0.3$ for class "earn" in Reuters and $\alpha = 0.1$ for class "A" in the letter recognition.) The reason that the performance of MC and SVMC is susceptible to a low α is that, as we explicated via Theorem 1 in Section 2.3, when the

Table 2. Performance results for different settings. $|P_U|$: No. of positives in U.

	F_1 (1st row), accuracy (2nd row)						
Settings	MC	SVMC	OSVM	SVM_NN	S-EM	NB_NN	ISVM
				"Earn" in Reuters			
$\alpha = 1.0, \beta = 0.7$	0.9513	0.9500	0.7768	0.0109	0.8120	0.8120	0.9931
	0.9923	0.9921	0.9595	0.9250	0.9709	0.9709	0.9990
$\alpha = 1.0, \beta = 0.5$	0.9427	0.9381	0.7432	0.0069	0.8033	0.8033	0.9877
	0.9927	0.9921	0.9605	0.9398	0.9747	0.9747	0.9985
$\alpha = 1.0, \beta = 0.3$	0.9017	0.9065	0.6426	0.0113	0.7491	0.7491	0.9914
	0.9919	0.9924	0.9656	0.9624	0.9779	0.9779	0.9994
$\alpha = 0.7, \beta = 1.0$	0.9646	0.9625	0.8412	0.0073	0.8146	0.8146	0.9855
	0.9922	0.9917	0.9626	0.8920	0.9621	0.9621	0.9969
$\alpha = 0.5, \beta = 1.0$	0.9639	0.9621	0.8468	0.0073	0.7900	0.7900	0.9822
	0.9921	0.9917	0.9674	0.8920	0.9583	0.9583	0.9962
$\alpha = 0.3, \beta = 1.0$	0.3667	0.2836	0.8763	0.0037	0.7664	0.7664	0.9774
	0.9149	0.9012	0.9736	0.8918	0.9559	0.9559	0.9768
				Letter "A"			
$\alpha = 1.0, \beta = 0.7$	0.9879	0.9679	0.0558	0.0826	0.6348	0.6348	0.9976
	0.9995	0.9987	0.9792	0.9795	0.9789	0.9789	0.9999
$\alpha = 1.0, \beta = 0.5$	0.9721	0.9295	0.0305	0.0889	0.2828	0.2828	0.9922
	0.9923	0.9983	0.9871	0.9875	0.9428	0.9428	0.9998
$\alpha = 1.0, \beta = 0.3$	0.9747	0.9342	0.0714	0.0200	0.1988	0.1988	0.9875
	0.9996	0.9990	0.9920	0.9926	0.9438	0.9438	0.9998
$\alpha = 0.7, \beta = 1.0$	0.9627	0.9685	0.0833	0.0632	0.4612	0.4612	0.9721
	0.9980	0.9983	0.9732	0.9729	0.9436	0.9436	0.9985
$\alpha = 0.5, \beta = 1.0$	0.9714	0.9577	0.0242	0.0811	0.6181	0.6181	0.9929
	0.9987	0.9981	0.9755	0.9726	0.9782	0.9782	0.9996
$\alpha = 0.3, \beta = 1.0$	0.9278	0.9507	0.0242	0.0242	0.6615	0.6615	0.9394
	0.9967	0.9977	0.9755	0.9755	0.9823	0.9823	0.9972
$\alpha = 0.1, \beta = 1.0$	0.3253	0.6065	0.0072	0.0072	0.6290	0.6290	0.8270
	0.9775	0.9842	0.9722	0.9722	0.9815	0.9815	0.9918



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

Figure 9. SVMC performance convergence on the "earn" class.

positive training data are seriously under-sampled, the final boundary of MC and SVMC would trespass the true boundary and result in fitting around the under-sampled data. In this case, the intermediate boundaries of MC or SVMC generated in the middle of the iterations give better results (see Figure 9(b)). We discuss this in more details in Section 4.4.3.

– β does not seem to influence the performance of MC and SVMC much 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. Note that, since a different β essentially defines a different test set, the performance for different β is not really comparable. Thus, a slightly low F_1 score of MC and SVMC for low β does not necessarily mean an inferior performance; it can be a result of an "easier" classification task due to lower amount of noise. The classification accuracy is very stable but not informative, since it is dominated by the large number of true negative examples in our test set (U).
– OSVM shows stable performance less affected by α or β. The stable performance is achieved by the unfairly performed optimization which is very hard to perform in high dimensional spaces. However, for low dimensional spaces, with the optimization method of Tax and Duin (2001), OSVM could give more stable performance that others when α is very low.
– 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. (Liu et al. (2002) gives an example of "the class of movies we may enjoy watching" as a dominant positive class where S-EM might outperform NB_NN.)

4.4.3. Performance convergence of SVMC. Figure 9(a) shows the SVMC performance convergence on the "earn" class when the positive data is not seriously under-sampled. 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 decrement of the recall. Our experiments on all other classes in Table 1 also show similar shapes of the convergence graph.

Figure 9(b) shows the SVMC performance convergence when the positive data are too much under-sampled. The convergence happens after 30 iterations, but the highest performance is actually achieved at 5th iteration. As discussed in Section 2.3, MC obviously "over-iterated" in this case. Determining the best number of iterations is a hard problem because an SCC method is assumed to have no labeled negative data available for training or optimization, which makes impossible to use existing validation methods to determine the performance dropping point. How to improve SVMC's performance with very few positive data is clearly an important direction for future research.

5. Conclusions and further work

We study Single-Class Classification (SCC) with positive and unlabeled data, which is a fundamental problem in various domains, such as text classification or pattern recognition. We present the MC algorithm which computes an accurate classification boundary of the positive data without relying on labeled negative data by iteratively applying a margin maximization classifier, such as SVM, to progressively obtain negative data from the unlabeled data. We also present the SVMC algorithm that optimizes the MC algorithm for fast training by removing the data, from the training set at each iteration, that are unlikely to be the support vectors. We evaluate MC and SVMC using various domains of real data sets. Experiment results show that with a reasonable amount of positive data, the MC and SVMC algorithm give the best performance among all the existing SCC methods that we compared with, and SVMC trains much faster than MC. When the positive training data is seriously under-sampled or no wide gaps exist between positive and negative classes, the boundary of MC or SVMC tend 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 SVMC given positive and unlabeled data.

Notes

1. http://www.csie.ntu.edu.tw/∼cjlin/libsvm
2. http://www.csie.ntu.edu.tw/∼cjlin/libsvm
3. http://www.cs.uic.edu/∼liub/S-EM/readme.html
4. http://www.daviddlewis.com/resources/testcollections/reuters21578
5. http://www.ics.uci.edu/mlearn/MLRepository.html
6. We used Rainbow (www-2.cs.cmu.edu/∼mccallum/bow/rainbow/) for text processing.
7. We ran our experiments on a linux machine of Pentium III 800 MHz with 384 MB memory.

References

Bileschi, S. M., & Heisele, B. (2003). Advances in component-based face detection. In *IEEE Int. Workshop on Analysis and Modeling of Faces and Gestures* (pp. 149–156).

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*, 121–167.

Chang, C.-C., & Lin, C.-J. (2001). Training nu-support vector classifiers: Theory and algorithms. *Neural Computation, 13*, 2119–2147.

Christianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

DeComite, F., Denis, F., & Gilleron, R. (1999). Positive and unlabeled examples help learning. In *Proc. Int. Conf. Algorithmic Learning Theory (ALT'99)* (pp. 219–230).

Denis, F. (1998). PAC Learning from positive statistical queries. In *Proc. Int. Conf. Algorithmic Learning Theory (ALT'99)* (pp. 112–126).

Frosini, A., Gori, M., & Priami, P. (1996). A neural network-based model for paper currency recognition and verification. *IEEE Transactions on Neural Networks, 7:6*, 1482–1490.

Gori, M., Lastrucci, L., & Soda, G. (1995). Autoassociator-based models for speaker verification. *Pattern Recognition Letters, 17*, 241–250.

Joachims, T. (2001). A statistical learning model of text classification with support vector machines. In *Proc. ACM SIGIR Int. Conf. Information Retrieval (SIGIR'01)* (pp. 128–136).

Letouzey, F., Denis, F., & Gilleron, R. (2000). Learning from positive and unlabeled examples. In *Proc. Int. Conf. Algorithmic Learning Theory (ALT'00)* (pp. 11–30).

Liu, B., Lee, W. S., Yu, P. S., & Li, X. (2002). Partially supervised classification of text documents. In *Proc. Int. Conf. Machine Learning (ICML'02)* (pp. 387–394).

Manevitz, L. M., & Yousef, M. (2001). One-class SVMs for document classification. *Journal of Machine Learning Research, 2*, 139–154.

Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation, B:7* 1443–1471.

Scholkopf, B., Smola, A. J., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation, 12*, 1083–1121.

Tax, D. M. J., & Duin, R. P. W. (2001). Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research, 2*, 155–173.

Tong, S. and Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Proc. Int. Conf. Machine Learning (ICML'00)* (pp. 999–1006).

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley and Sons.

Yu, H. (2003). SVMC: Single-class classification with support vector machines. In *Proc. Int. Joint Conf. on Articial Intelligence (IJCAI'03)* (pp. 567–572).

Yu, H., Han, J., & Chang, K. C. (2002). PEBL: Positive-example based learning for web page classification using SVM. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'02)* (pp. 239–248).

Received December 5, 2003
Revised March 10, 2005
Accepted March 11, 2005