Journal of Machine Learning Research (2001) 45-66 Submitted 10/01; Published 11/01

Support Vector Machine Active Learning with Applications to Text Classification

Simon Tong SIMON.TONG@CS.STANFORD.EDU
Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford CA 94305-9010, USA

Editor: Leslie Pack Kaelbling

Abstract

Support vector machines have met with significant success in numerous real-world learning tasks. However, like most machine learning algorithms, they are generally applied using a randomly selected training set classified in advance. In many settings, we also have the option of using *pool-based active learning*. Instead of using a randomly selected training set, the learner has access to a pool of unlabeled instances and can request the labels for some number of them. We introduce a new algorithm for performing active learning with support vector machines, i.e., an algorithm for choosing which instances to request next. We provide a theoretical motivation for the algorithm using the notion of a *version space*. We present experimental results showing that employing our active learning method can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.

Keywords: Active Learning, Selective Sampling, Support Vector Machines, Classification, Relevance Feedback

1. Introduction

In many supervised learning tasks, labeling instances to create a training set is time-consuming and costly; thus, finding ways to minimize the number of labeled instances is beneficial. Usually, the training set is chosen to be a random sampling of instances. However, in many cases *active learning* can be employed. Here, the learner can actively choose the training data. It is hoped that allowing the learner this extra flexibility will reduce the learner's need for large quantities of labeled data.

Pool-based active learning for classification was introduced by Lewis and Gale (1994). The learner has access to a pool of unlabeled data and can request the true class label for a certain number of instances in the pool. In many domains this is a reasonable approach since a large quantity of unlabeled data is readily available. The main issue with active learning is finding a way to choose good requests or *queries* from the pool.

Examples of situations in which pool-based active learning can be employed are:

- **Web searching.** A Web-based company wishes to search the web for particular types of pages (e.g., pages containing lists of journal publications). It employs a number of people to hand-label some web pages so as to create a training set for an automatic classifier that will eventually be used to classify the rest of the web. Since human expertise is a limited resource, the company wishes to reduce the number of pages the employees have to label. Rather than labeling pages randomly drawn from the web, the computer requests targeted pages that it believes will be most informative to label.

- **Email filtering.** The user wishes to create a personalized automatic junk email filter. In the learning phase the automatic learner has access to the user's past email files. It interactively brings up past email and asks the user whether the displayed email is junk mail or not. Based on the user's answer it brings up another email and queries the user. The process is repeated some number of times and the result is an email filter tailored to that specific person.

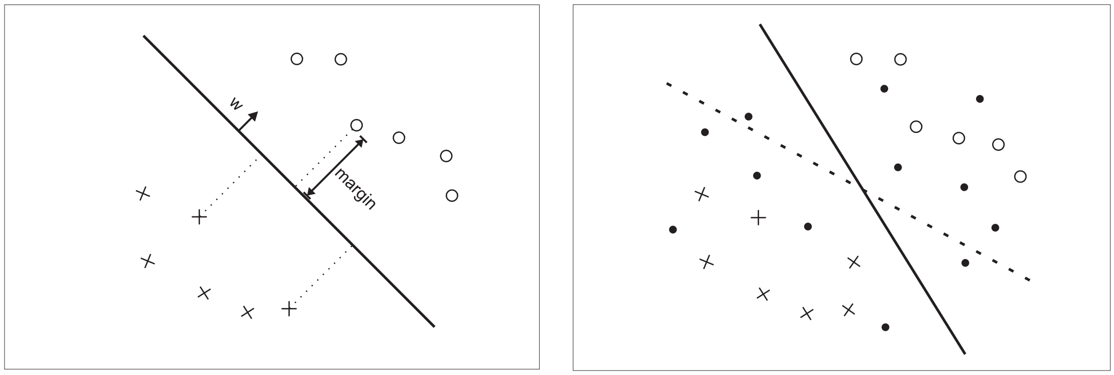
- **Relevance feedback.** The user wishes to sort through a database or website for items (images, articles, etc.) that are of personal interest—an "I'll know it when I see it" type of search. The computer displays an item and the user tells the learner whether the item is interesting or not. Based on the user's answer, the learner brings up another item from the database. After some number of queries the learner then returns a number of items in the database that it believes will be of interest to the user.

The first two examples involve *induction*. The goal is to create a classifier that works well on unseen future instances. The third example is an example of *transduction*(Vapnik, 1998). The learner's performance is assessed on the remaining instances in the database rather than a totally independent test set.

We present a new algorithm that performs pool-based active learning with support vector machines (SVMs). We provide theoretical motivations for our approach to choosing the queries, together with experimental results showing that active learning with SVMs can significantly reduce the need for labeled training instances.

We shall use text classification as a running example throughout this paper. This is the task of determining to which pre-defined topic a given text document belongs. Text classification has an important role to play, especially with the recent explosion of readily available text data. There have been many approaches to achieve this goal (Rocchio, 1971, Dumais et al., 1998, Sebastiani, 2001). Furthermore, it is also a domain in which SVMs have shown notable success (Joachims, 1998, Dumais et al., 1998) and it is of interest to see whether active learning can offer further improvement over this already highly effective method.

The remainder of the paper is structured as follows. Section 2 discusses the use of SVMs both in terms of induction and transduction. Section 3 then introduces the notion of a *version space* and Section 4 provides theoretical motivation for three methods for performing active learning with SVMs. In Section 5 we present experimental results for two real-world text domains that indicate that active learning can significantly reduce the need for labeled instances in practice. We conclude in Section 7 with some discussion of the potential significance of our results and some directions for future work.



(a) (b)

Figure 1: (a) A simple linear support vector machine. (b) A SVM (dotted line) and a transductive SVM (solid line). Solid circles represent unlabeled instances.

2. Support Vector Machines

Support vector machines (Vapnik, 1982) have strong theoretical foundations and excellent empirical successes. They have been applied to tasks such as handwritten digit recognition, object recognition, and text classification.

2.1 SVMs for Induction

We shall consider SVMs in the binary classification setting. We are given training data $\{\mathbf{x}_1 \dots \mathbf{x}_n\}$ that are vectors in some space $\mathcal{X} \subseteq \mathbb{R}^d$. We are also given their labels $\{y_1 \dots y_n\}$ where $y_i \in \{-1, 1\}$. In their simplest form, SVMs are hyperplanes that separate the training data by a maximal margin (see Fig. 1a) . All vectors lying on one side of the hyperplane are labeled as -1, and all vectors lying on the other side are labeled as 1. The training instances that lie closest to the hyperplane are called *support vectors*. More generally, SVMs allow one to project the original training data in space $\mathcal{X}$ to a higher dimensional feature space $\mathcal{F}$ via a Mercer kernel operator K. In other words, we consider the set of classifiers of the form:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \right). \tag{1}$$

When K satisfies Mercer's condition (Burges, 1998) we can write: $K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$ where $\Phi : \mathcal{X} \to \mathcal{F}$ and "$\cdot$" denotes an inner product. We can then rewrite f as:

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}), \text{ where } \mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i). \tag{2}$$

Thus, by using K we are implicitly projecting the training data into a different (often higher dimensional) feature space $\mathcal{F}$. The SVM then computes the α_is that correspond to the maximal margin hyperplane in $\mathcal{F}$. By choosing different kernel functions we can implicitly project the training data from $\mathcal{X}$ into spaces $\mathcal{F}$ for which hyperplanes in $\mathcal{F}$ correspond to more complex decision boundaries in the original space $\mathcal{X}$.

Two commonly used kernels are the polynomial kernel given by $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^p$ which induces polynomial boundaries of degree p in the original space $\mathcal{X}$[1] and the radial basis function kernel $K(\mathbf{u}, \mathbf{v}) = (e^{-\gamma(\mathbf{u}-\mathbf{v})\cdot(\mathbf{u}-\mathbf{v})})$ which induces boundaries by placing weighted Gaussians upon key training instances. For the majority of this paper we will assume that the modulus of the training data feature vectors are constant, i.e., for all training instances $\mathbf{x}_i$, $\|\Phi(\mathbf{x}_i)\| = \lambda$ for some fixed λ. The quantity $\|\Phi(\mathbf{x}_i)\|$ is always constant for radial basis function kernels, and so the assumption has no effect for this kernel. For $\|\Phi(\mathbf{x}_i)\|$ to be constant with the polynomial kernels we require that $\|\mathbf{x}_i\|$ be constant. It is possible to relax this constraint on $\Phi(\mathbf{x}_i)$ and we shall discuss this at the end of Section 4.

2.2 SVMs for Transduction

The previous subsection worked within the framework of *induction*. There was a labeled training set of data and the task was to create a classifier that would have good performance on *unseen* test data. In addition to regular induction, SVMs can also be used for *transduction*. Here we are first given a set of both labeled and unlabeled data. The learning task is to assign labels to the unlabeled data as accurately as possible. SVMs can perform transduction by finding the hyperplane that maximizes the margin relative to both the labeled and unlabeled data. See Figure 1b for an example. Recently, *transductive SVMs* (TSVMs) have been used for text classification (Joachims, 1999b), attaining some improvements in precision/recall breakeven performance over regular inductive SVMs.

3. Version Space

Given a set of labeled training data and a Mercer kernel K, there is a set of hyperplanes that separate the data in the induced feature space $\mathcal{F}$. We call this set of consistent hypotheses the *version space* (Mitchell, 1982). In other words, hypothesis f is in version space if for every training instance $\mathbf{x}_i$ with label y_i we have that $f(\mathbf{x}_i) > 0$ if $y_i = 1$ and $f(\mathbf{x}_i) < 0$ if $y_i = -1$. More formally:

Definition 1 *Our set of possible hypotheses is given as:*

$$\mathcal{H} = \left\{ f \mid f(\mathbf{x}) = \frac{\mathbf{w} \cdot \Phi(\mathbf{x})}{\|\mathbf{w}\|} \text{ where } \mathbf{w} \in \mathcal{W} \right\},$$

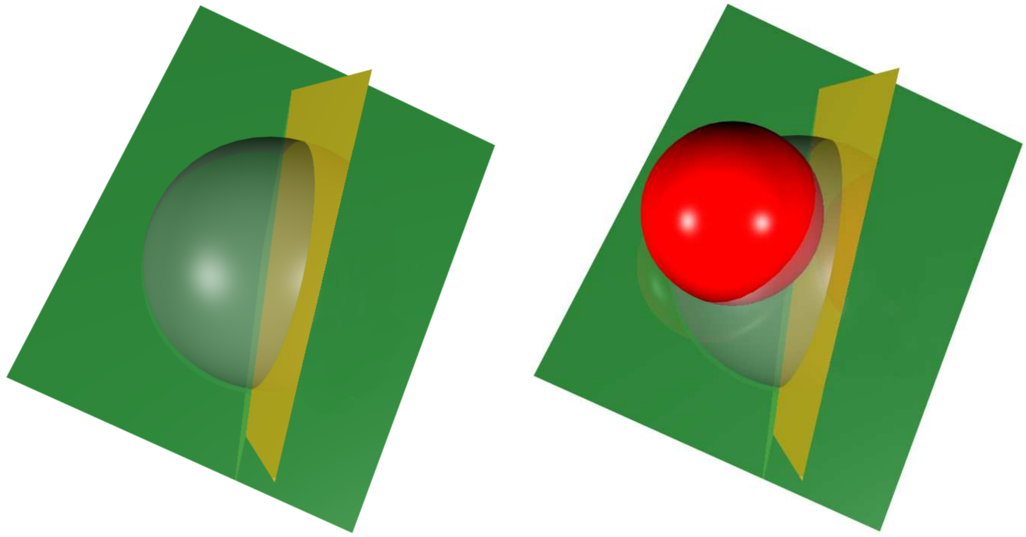
where our parameter space $\mathcal{W}$ *is simply equal to* $\mathcal{F}$. *The* version space, $\mathcal{V}$ *is then defined as:*

$$\mathcal{V} = \{f \in \mathcal{H} \mid \forall i \in \{1 \ldots n\} \;\; y_i f(\mathbf{x}_i) > 0\}.$$

Notice that since $\mathcal{H}$ is a set of hyperplanes, there is a bijection between unit vectors $\mathbf{w}$ *and hypotheses f in $\mathcal{H}$. Thus we will redefine $\mathcal{V}$ as:*

$$\mathcal{V} = \{\mathbf{w} \in \mathcal{W} \mid \|\mathbf{w}\| = 1, \; y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0, i = 1 \ldots n\}.$$

1. We have not introduced a bias weight in Eq. (2). Thus, the simple Euclidean inner product will produce hyperplanes that pass through the origin. However, a polynomial kernel of degree one induces hyperplanes that do not need to pass through the origin.



(a) (b)

Figure 2: (a) Version space duality. The surface of the hypersphere represents unit weight vectors. Each of the two hyperplanes corresponds to a labeled training instance. Each hyperplane restricts the area on the hypersphere in which consistent hypotheses can lie. Here, the version space is the surface segment of the hypersphere closest to the camera. (b) An SVM classifier in a version space. The dark embedded sphere is the largest radius sphere whose center lies in the version space and whose surface does not intersect with the hyperplanes. The center of the embedded sphere corresponds to the SVM, its radius is proportional to the margin of the SVM in $\mathcal{F}$, and the training points corresponding to the hyperplanes that it touches are the support vectors.

Note that a version space only exists if the *training* data are linearly separable in the feature space. Thus, we require linear separability of the training data in the feature space. This restriction is much less harsh than it might at first seem. First, the feature space often has a very high dimension and so in many cases it results in the data set being linearly separable. Second, as noted by Shawe-Taylor and Cristianini (1999), it is possible to modify any kernel so that the data in the new induced feature space is linearly separable[2].

There exists a duality between the feature space $\mathcal{F}$ and the parameter space $\mathcal{W}$ (Vapnik, 1998, Herbrich et al., 2001) which we shall take advantage of in the next section: points in $\mathcal{F}$ correspond to hyperplanes in $\mathcal{W}$ and *vice versa*.

By definition, points in $\mathcal{W}$ correspond to hyperplanes in $\mathcal{F}$. The intuition behind the converse is that observing a training instance $\mathbf{x}_i$ in the feature space restricts the set of separating hyperplanes to ones that classify $\mathbf{x}_i$ correctly. In fact, we can show that the set

2. This is done by redefining for all training instances $\mathbf{x}_i$: $K(\mathbf{x}_i, \mathbf{x}_i) \leftarrow K(\mathbf{x}_i, \mathbf{x}_i) + \nu$ where ν is a positive regularization constant. This essentially achieves the same effect as the soft margin error function (Cortes and Vapnik, 1995) commonly used in SVMs. It permits the training data to be linearly non-separable in the original feature space.

of allowable points $\mathbf{w}$ in $\mathcal{W}$ is restricted to lie on one side of a hyperplane in $\mathcal{W}$. More formally, to show that points in $\mathcal{F}$ correspond to hyperplanes in $\mathcal{W}$, suppose we are given a new training instance $\mathbf{x}_i$ with label y_i. Then any separating hyperplane must satisfy $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0$. Now, instead of viewing $\mathbf{w}$ as the normal vector of a hyperplane in $\mathcal{F}$, think of $\Phi(\mathbf{x}_i)$ as being the normal vector of a hyperplane in $\mathcal{W}$. Thus $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0$ defines a half space in $\mathcal{W}$. Furthermore $\mathbf{w} \cdot \Phi(\mathbf{x}_i) = 0$ defines a hyperplane in $\mathcal{W}$ that acts as one of the boundaries to version space $\mathcal{V}$. Notice that the version space is a connected region on the surface of a hypersphere in parameter space. See Figure 2a for an example.

SVMs find the hyperplane that maximizes the margin in the feature space $\mathcal{F}$. One way to pose this optimization task is as follows:

$$\begin{array}{cc} \text{maximize}_{\mathbf{w} \in \mathcal{F}} & \min_i \{y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i))\} \\ \text{subject to:} & \|\mathbf{w}\| = 1 \\ & y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0 \quad i = 1 \dots n. \end{array}$$

By having the conditions $\|\mathbf{w}\| = 1$ and $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0$ we cause the solution to lie in the version space. Now, we can view the above problem as finding the point $\mathbf{w}$ in the version space that maximizes the distance: $\min_i \{y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i))\}$. From the duality between feature and parameter space, and since $\|\Phi(\mathbf{x}_i)\| = \lambda$, each $\Phi(\mathbf{x}_i)/\lambda$ is a unit normal vector of a hyperplane in parameter space. Because of the constraints $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0 \quad i = 1 \dots n$ each of these hyperplanes delimit the version space. The expression $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i))$ can be regarded as:

$$\lambda \times \text{ the distance between the point } \mathbf{w} \text{ and the hyperplane with normal vector } \Phi(\mathbf{x}_i).$$

Thus, we want to find the point $\mathbf{w}^*$ in the version space that maximizes the minimum distance to any of the delineating hyperplanes. That is, SVMs find the center of the largest radius hypersphere whose center can be placed in the version space and whose surface does not intersect with the hyperplanes corresponding to the labeled instances, as in Figure 2b.

The normals of the hyperplanes that are touched by the maximal radius hypersphere are the $\Phi(\mathbf{x}_i)$ for which the distance $y_i(\mathbf{w}^* \cdot \Phi(\mathbf{x}_i))$ is minimal. Now, taking the original rather than the dual view, and regarding $\mathbf{w}^*$ as the unit normal vector of the SVM and $\Phi(\mathbf{x}_i)$ as points in feature space, we see that the hyperplanes that are touched by the maximal radius hypersphere correspond to the support vectors (i.e., the labeled points that are closest to the SVM hyperplane boundary).

The radius of the sphere is the distance from the center of the sphere to one of the touching hyperplanes and is given by $y_i(\mathbf{w}^* \cdot \Phi(\mathbf{x}_i)/\lambda)$ where $\Phi(\mathbf{x}_i)$ is a support vector. Now, viewing $\mathbf{w}^*$ as a unit normal vector of the SVM and $\Phi(\mathbf{x}_i)$ as points in feature space, we have that the distance $y_i(\mathbf{w}^* \cdot \Phi(\mathbf{x}_i)/\lambda)$ is:

$$\frac{1}{\lambda} \times \text{ the distance between support vector } \Phi(\mathbf{x}_i) \text{ and the hyperplane with normal vector } \mathbf{w},$$

which is the margin of the SVM divided by λ. Thus, the radius of the sphere is proportional to the margin of the SVM.

4. Active Learning

In pool-based active learning we have a pool of unlabeled instances. It is assumed that the instances $\mathbf{x}$ are independently and identically distributed according to some underlying distribution $F(\mathbf{x})$ and the labels are distributed according to some conditional distribution $P(y \mid \mathbf{x})$.

Given an unlabeled pool U, an *active learner* ℓ has three components: (f, q, X). The first component is a classifier, $f : \mathcal{X} \to \{-1, 1\}$, trained on the current set of labeled data X (and possibly unlabeled instances in U too). The second component $q(X)$ is the querying function that, given a current labeled set X, decides which instance in U to query next. The active learner can return a classifier f after each query (*online learning*) or after some fixed number of queries.

The main difference between an active learner and a passive learner is the querying component q. This brings us to the issue of how to choose the next unlabeled instance to query. Similar to Seung et al. (1992), we use an approach that queries points so as to attempt to reduce the size of the version space as much as possible. We take a myopic approach that greedily chooses the next query based on this criterion. We also note that myopia is a standard approximation used in sequential decision making problems Horvitz and Rutledge (1991), Latombe (1991), Heckerman et al. (1994). We need two more definitions before we can proceed:

Definition 2 *Area($\mathcal{V}$) is the surface area that the version space $\mathcal{V}$ occupies on the hypersphere $\|\mathbf{w}\| = 1$.*

Definition 3 *Given an active learner ℓ, let $\mathcal{V}_i$ denote the version space of ℓ after i queries have been made. Now, given the $(i+1)$th query $\mathbf{x}_{i+1}$, define:*

$$\begin{aligned}
\mathcal{V}_i^- &= \mathcal{V}_i \cap \{\mathbf{w} \in \mathcal{W} \mid -(\mathbf{w} \cdot \Phi(\mathbf{x}_{i+1})) > 0\}, \\
\mathcal{V}_i^+ &= \mathcal{V}_i \cap \{\mathbf{w} \in \mathcal{W} \mid +(\mathbf{w} \cdot \Phi(\mathbf{x}_{i+1})) > 0\}.
\end{aligned}$$

So $\mathcal{V}_i^-$ and $\mathcal{V}_i^+$ denote the resulting version spaces when the next query $\mathbf{x}_{i+1}$ is labeled as -1 and 1 respectively.

We wish to reduce the version space as fast as possible. Intuitively, one good way of doing this is to choose a query that halves the version space. The follow lemma says that, for any given number of queries, the learner that chooses successive queries that halves the version spaces is the learner that minimizes the maximum expected size of the version space, where the maximum is taken over all conditional distributions of y given $\mathbf{x}$:

Lemma 4 *Suppose we have an input space $\mathcal{X}$, finite dimensional feature space $\mathcal{F}$ (induced via a kernel K), and parameter space $\mathcal{W}$. Suppose active learner ℓ^* always queries instances whose corresponding hyperplanes in parameter space $\mathcal{W}$ halves the area of the current version space. Let ℓ be any other active learner. Denote the version spaces of ℓ^* and ℓ after i queries as $\mathcal{V}_i^*$ and $\mathcal{V}_i$ respectively. Let $\mathcal{P}$ denote the set of all conditional distributions of y given $\mathbf{x}$. Then,*

$$\forall i \in \mathbb{N}^+ \ \sup_{P \in \mathcal{P}} E_P[Area(\mathcal{V}_i^*)] \leq \sup_{P \in \mathcal{P}} E_P[Area(\mathcal{V}_i)],$$

with strict inequality whenever there exists a query $j \in \{1 \ldots i\}$ by ℓ that does not halve version space V_{j-1}.

Proof. The proof is straightforward. The learner, ℓ^* always chooses to query instances that halve the version space. Thus $Area(\mathcal{V}_{i+1}^*) = \frac{1}{2} Area(\mathcal{V}_i^*)$ no matter what the labeling of the query points are. Let r denote the dimension of feature space $\mathcal{F}$. Then r is also the dimension of the parameter space $\mathcal{W}$. Let S_r denote the surface area of the unit hypersphere of dimension r. Then, under any conditional distribution P, $Area(\mathcal{V}_i^*) = S_r/2^i$.

Now, suppose ℓ does not always query an instance that halves the area of the version space. Then after some number, k, of queries ℓ first chooses to query a point $\mathbf{x}_{k+1}$ that does not halve the current version space $\mathcal{V}_k$. Let $y_{k+1} \in \{-1, 1\}$ correspond to the labeling of $\mathbf{x}_{k+1}$ that will cause the larger half of the version space to be chosen.

Without loss of generality assume $Area(\mathcal{V}_k^-) > Area(\mathcal{V}_k^+)$ and so $y_{k+1} = -1$. Note that $Area(\mathcal{V}_k^-) + Area(\mathcal{V}_k^+) = S_r/2^k$, so we have that $Area(\mathcal{V}_k^-) > S_r/2^{k+1}$.

Now consider the conditional distribution P_0:

$$P_0(-1 \mid \mathbf{x}) = \begin{cases} \frac{1}{2} & \text{if } \mathbf{x} \neq \mathbf{x}_{k+1} \\ 1 & \text{if } \mathbf{x} = \mathbf{x}_{k+1} \end{cases}.$$

Then under this distribution, $\forall i > k$,

$$E_{P_0}[Area(\mathcal{V}_i)] = \frac{1}{2^{i-k-1}} Area(\mathcal{V}_k^-) > \frac{S_r}{2^i}.$$

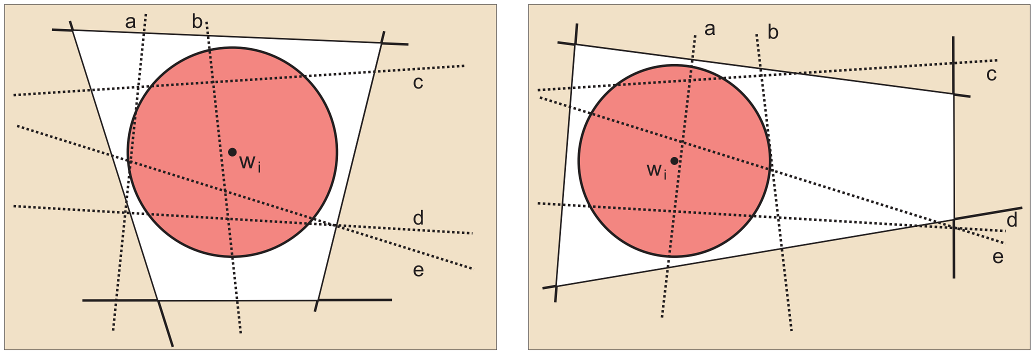
Hence, $\forall i > k$,

$$\sup_{P \in \mathcal{P}} E_P[Area(\mathcal{V}_i^*)] > \sup_{P \in \mathcal{P}} E_P[Area(\mathcal{V}_i)].$$

□

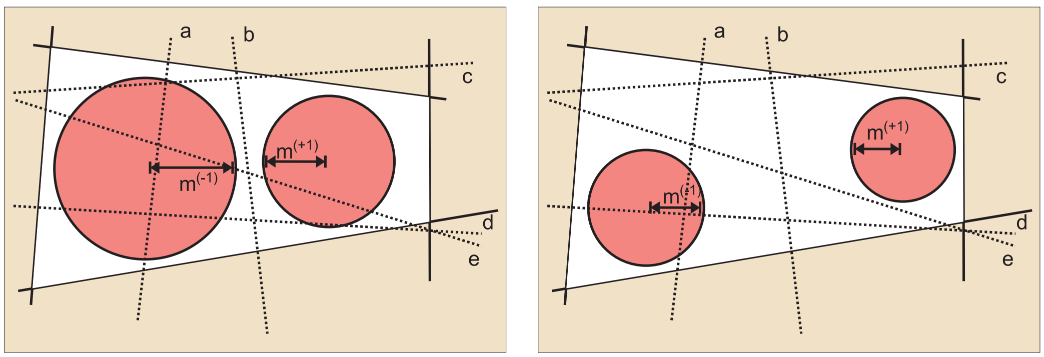
Now, suppose $\mathbf{w}^* \in \mathcal{W}$ is the unit parameter vector corresponding to the SVM that we would have obtained had we known the actual labels of *all* of the data in the pool. We know that $\mathbf{w}^*$ must lie in each of the version spaces $\mathcal{V}_1 \supset \mathcal{V}_2 \supset \mathcal{V}_3 \ldots$, where $\mathcal{V}_i$ denotes the version space after i queries. Thus, by shrinking the size of the version space as much as possible with each query, we are reducing as fast as possible the space in which $\mathbf{w}^*$ can lie. Hence, the SVM that we learn from our limited number of queries will lie close to $\mathbf{w}^*$.

If one is willing to assume that there is a hypothesis lying within $\mathcal{H}$ that generates the data and that the generating hypothesis is deterministic and that the data are noise free, then strong generalization performance properties of an algorithm that halves version space can also be shown (Freund et al., 1997). For example one can show that the generalization error decreases exponentially with the number of queries.



(a) (b)

Figure 3: (a) Simple Margin will query **b**. (b) Simple Margin will query **a**.



(a) (b)

Figure 4: (a) MaxMin Margin will query **b**. The two SVMs with margins m^- and m^+ for **b** are shown. (b) Ratio Margin will query **e**. The two SVMs with margins m^- and m^+ for **e** are shown.

This discussion provides motivation for an approach where we query instances that split the current version space into two equal parts as much as possible. Given an unlabeled instance $\mathbf{x}$ from the pool, it is not practical to explicitly compute the sizes of the new version spaces $\mathcal{V}^-$ and $\mathcal{V}^+$ (i.e., the version spaces obtained when $\mathbf{x}$ is labeled as -1 and $+1$ respectively). We next present three ways of approximating this procedure.

- **Simple Margin.** Recall from section 3 that, given some data $\{\mathbf{x}_1 \ldots \mathbf{x}_i\}$ and labels $\{y_1 \ldots y_i\}$, the SVM unit vector $\mathbf{w}_i$ obtained from this data is the center of the largest hypersphere that can fit inside the current version space $\mathcal{V}_i$. The position of $\mathbf{w}_i$ in the version space $\mathcal{V}_i$ clearly depends on the shape of the region $\mathcal{V}_i$, however it is often approximately in the center of the version space. Now, we can test each of the unlabeled instances $\mathbf{x}$ in the pool to see how close their corresponding hyperplanes in $\mathcal{W}$ come to the centrally placed $\mathbf{w}_i$. The closer a hyperplane in $\mathcal{W}$ is to the point $\mathbf{w}_i$, the more centrally it is placed in the version space, and the more it bisects the version space. Thus we can pick the unlabeled instance in the pool whose hyperplane in $\mathcal{W}$ comes closest to the vector $\mathbf{w}_i$. For each unlabeled instance $\mathbf{x}$, the shortest distance between its hyperplane in $\mathcal{W}$ and the vector $\mathbf{w}_i$ is simply the distance between the feature vector $\Phi(\mathbf{x})$ and the hyperplane $\mathbf{w}_i$ in $\mathcal{F}$—which is easily computed by $|\mathbf{w}_i \cdot \Phi(\mathbf{x})|$. This results in the natural rule: learn an SVM on the existing labeled data and choose as the next instance to query the instance that comes closest to the hyperplane in $\mathcal{F}$.

Figure 3a presents an illustration. In the stylized picture we have flattened out the surface of the unit weight vector hypersphere that appears in Figure 2a. The white area is version space $\mathcal{V}_i$ which is bounded by solid lines corresponding to labeled instances. The five dotted lines represent unlabeled instances in the pool. The circle represents the largest radius hypersphere that can fit in the version space. Note that the edges of the circle do not touch the solid lines—just as the dark sphere in 2b does not meet the hyperplanes on the surface of the larger hypersphere (they meet somewhere under the surface). The instance $\mathbf{b}$ is closest to the SVM $\mathbf{w}_i$ and so we will choose to query $\mathbf{b}$.

- **MaxMin Margin.** The Simple Margin method can be a rather rough approximation. It relies on the assumption that the version space is fairly symmetric and that $\mathbf{w}_i$ is centrally placed. It has been demonstrated, both in theory and practice, that these assumptions can fail significantly (Herbrich et al., 2001). Indeed, if we are not careful we may actually query an instance whose hyperplane does not even intersect the version space. The MaxMin approximation is designed to overcome these problems to some degree. Given some data $\{\mathbf{x}_1 \ldots \mathbf{x}_i\}$ and labels $\{y_1 \ldots y_i\}$, the SVM unit vector $\mathbf{w}_i$ is the center of the largest hypersphere that can fit inside the current version space $\mathcal{V}_i$ and the radius m_i of the hypersphere is proportional[3] to the size of the margin of $\mathbf{w}_i$. We can use the radius m_i as an indication of the size of the version space (Vapnik, 1998). Suppose we have a candidate unlabeled instance $\mathbf{x}$ in the pool. We can estimate the relative size of the resulting version space $\mathcal{V}^-$ by labeling $\mathbf{x}$ as -1, finding the SVM obtained from adding $\mathbf{x}$ to our labeled training data and looking at the size of its margin m^-. We can perform a similar calculation for $\mathcal{V}^+$ by relabeling $\mathbf{x}$ as class $+1$ and finding the resulting SVM to obtain margin m^+.

  Since we want an equal split of the version space, we wish $Area(\mathcal{V}^-)$ and $Area(\mathcal{V}^+)$ to be similar. Now, consider $\min(Area(\mathcal{V}^-), Area(\mathcal{V}^+))$. It will be small if $Area(\mathcal{V}^-)$ and $Area(\mathcal{V}^+)$ are very different. Thus we will consider $\min(m^-, m^+)$ as an approximation and we will choose to query the $\mathbf{x}$ for which this quantity is largest. Hence, the MaxMin query algorithm is as follows: for each unlabeled instance $\mathbf{x}$ compute the margins m^- and m^+ of the SVMs obtained when we label $\mathbf{x}$ as -1 and $+1$ respectively; then choose to query the unlabeled instance for which the quantity $\min(m^-, m^+)$ is greatest.

  Figures 3b and 4a show an example comparing the Simple Margin and MaxMin Margin methods.

- **Ratio Margin.** This method is similar in spirit to the MaxMin Margin method. We use m^- and m^+ as indications of the sizes of $\mathcal{V}^-$ and $\mathcal{V}^+$. However, we shall try to

3. To ease notation, without loss of generality we shall assume the the constant of proportionality is 1, i.e., the radius is equal to the margin.

take into account the fact that the current version space $\mathcal{V}_i$ may be quite elongated and for some $\mathbf{x}$ in the pool *both* m^- and m^+ may be small simply because of the shape of version space. Thus we will instead look at the *relative* sizes of m^- and m^+ and choose to query the $\mathbf{x}$ for which $\min(\frac{m^-}{m^+}, \frac{m^+}{m^-})$ is largest (see Figure 4b).

The above three methods are approximations to the querying component that always halves version space. After performing some number of queries we then return a classifier by learning a SVM with the labeled instances.

The margin can be used as an indication of the version space size irrespective of whether the feature vectors have constant modulus. Thus the explanation for the MaxMin and Ratio methods still holds even without the constraint on the modulus of the training feature vectors. The Simple method can still be used when the training feature vectors do not have constant modulus, but the motivating explanation no longer holds since the maximal margin hyperplane can no longer be viewed as the center of the largest allowable sphere. However, for the Simple method, alternative motivations have recently been proposed by Campbell et al. (2000) that do not require the constraint on the modulus.

For inductive learning, after performing some number of queries we then return a classifier by learning a SVM with the labeled instances. For transductive learning, after querying some number of instances we then return a classifier by learning a transductive SVM with the labeled *and* unlabeled instances.

5. Experiments

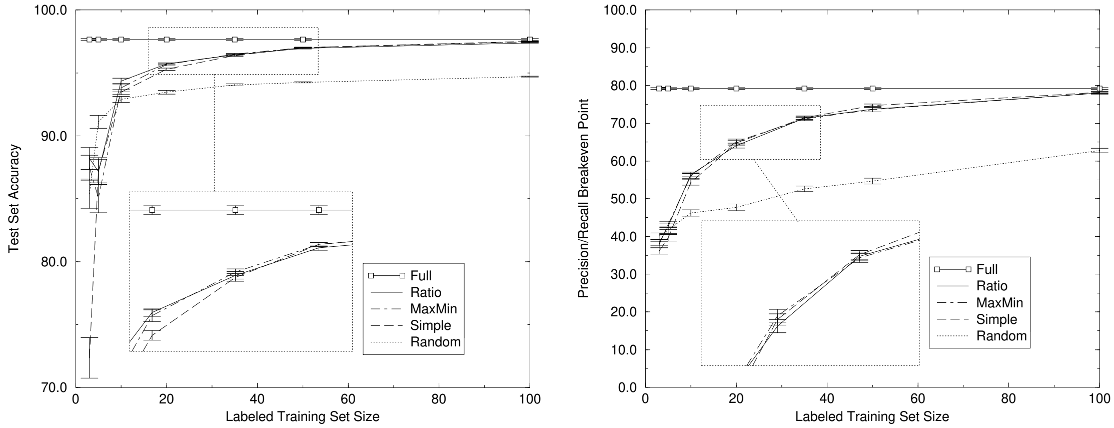
For our empirical evaluation of the above methods we used two real-world text classification domains: the Reuters-21578 data set and the Newsgroups data set.

5.1 Reuters Data Collection Experiments

The Reuters-21578 data set[4] is a commonly used collection of newswire stories categorized into hand-labeled topics. Each news story has been hand-labeled with some number of topic labels such as "corn", "wheat" and "corporate acquisitions". Note that some of the topics overlap and so some articles belong to more than one category. We used the 12902 articles from the "ModApte" split of the data[5] and, to stay comparable with previous studies, we considered the top ten most frequently occurring topics. We learned ten different binary classifiers, one to distinguish each topic. Each document was represented as a stemmed, TFIDF-weighted word frequency vector.[6] Each vector had unit modulus. A stop list of common words was used and words occurring in fewer than three documents were also ignored. Using this representation, the document vectors had about 10000 dimensions.

We first compared the three querying methods in the inductive learning setting. Our test set consisted of the 3299 documents present in the "ModApte" test set.

4. Obtained from www.research.att.com/~lewis.
5. The Reuters-21578 collection comes with a set of predefined training and test set splits. The commonly used"ModApte" split filters out duplicate articles and those without a labeled topic, and then uses earlier articles as the training set and later articles as the test set.
6. We used Rainbow (McCallum, 1996) for text processing.



(a) (b)

Figure 5: (a) Average test set accuracy over the ten most frequently occurring topics when using a pool size of 1000. (b) Average test set precision/recall breakeven point over the ten most frequently occurring topics when using a pool size of 1000.

Topic	Simple	MaxMin	Ratio	Equivalent Random size
Earn	86.39 ± 1.65	87.75 ± 1.40	90.24 ± 2.31	34
Acq	77.04 ± 1.17	77.08 ± 2.00	80.42 ± 1.50	> 100
Money-fx	93.82 ± 0.35	$\mathbf{94.80 \pm 0.14}$	$\mathbf{94.83 \pm 0.13}$	50
Grain	95.53 ± 0.09	95.29 ± 0.38	95.55 ± 1.22	13
Crude	95.26 ± 0.38	95.26 ± 0.15	95.35 ± 0.21	> 100
Trade	96.31 ± 0.28	96.64 ± 0.10	96.60 ± 0.15	> 100
Interest	96.15 ± 0.21	96.55 ± 0.09	96.43 ± 0.09	> 100
Ship	97.75 ± 0.11	97.81 ± 0.09	97.66 ± 0.12	> 100
Wheat	98.10 ± 0.24	98.48 ± 0.09	98.13 ± 0.20	> 100
Corn	98.31 ± 0.19	98.56 ± 0.05	98.30 ± 0.19	15

Table 1: Average test set accuracy over the top ten most frequently occurring topics (most frequent topic first) when trained with ten labeled documents. Boldface indicates statistical significance.

For each of the ten topics we performed the following steps. We created a pool of unlabeled data by sampling 1000 documents from the remaining data and removing their labels. We then randomly selected two documents in the pool to give as the initial labeled training set. One document was about the desired topic, and the other document was not about the topic. Thus we gave each learner 998 unlabeled documents and 2 labeled documents. After a fixed number of queries we asked each learner to return a classifier (an

Topic	Simple	MaxMin	Ratio	Equivalent Random size
Earn	86.05 ± 0.61	$\mathbf{89.03 \pm 0.53}$	$\mathbf{88.95 \pm 0.74}$	12
Acq	54.14 ± 1.31	56.43 ± 1.40	57.25 ± 1.61	12
Money-fx	35.62 ± 2.34	38.83 ± 2.78	38.27 ± 2.44	52
Grain	50.25 ± 2.72	$\mathbf{58.19 \pm 2.04}$	$\mathbf{60.34 \pm 1.61}$	51
Crude	58.22 ± 3.15	55.52 ± 2.42	58.41 ± 2.39	55
Trade	50.71 ± 2.61	48.78 ± 2.61	50.57 ± 1.95	85
Interest	40.61 ± 2.42	45.95 ± 2.61	43.71 ± 2.07	60
Ship	53.93 ± 2.63	52.73 ± 2.95	53.75 ± 2.85	> 100
Wheat	64.13 ± 2.10	66.71 ± 1.65	66.57 ± 1.37	> 100
Corn	49.52 ± 2.12	48.04 ± 2.01	46.25 ± 2.18	> 100

Table 2: Average test set precision/recall breakeven point over the top ten most frequently occurring topics (most frequent topic first) when trained with ten labeled documents. Boldface indicates statistical significance.

SVM with a polynomial kernel of degree one[7] learned on the labeled training documents). We then tested the classifier on the independent test set.

The above procedure was repeated thirty times for each topic and the results were averaged. We considered the Simple Margin, MaxMin Margin and Ratio Margin querying methods as well as a Random Sample method. The Random Sample method simply randomly chooses the next query point from the unlabeled pool. This last method reflects what happens in the regular passive learning setting—the training set is a random sampling of the data.

To measure performance we used two metrics: test set classification error and, to stay compatible with previous Reuters corpus results, the *precision/recall breakeven point* (Joachims, 1998). *Precision* is the percentage of documents a classifier labels as "relevant" that are really relevant. *Recall* is the percentage of relevant documents that are labeled as "relevant" by the classifier. By altering the decision threshold on the SVM we can trade precision for recall and can obtain a precision/recall curve for the test set. The precision/recall breakeven point is a one number summary of this graph: it is the point at which precision equals recall.

Figures 5a and 5b present the average test set accuracy and precision/recall breakeven points over the ten topics as we vary the number of queries permitted. The horizontal line is the performance level achieved when the SVM is trained on all 1000 labeled documents comprising the pool. Over the Reuters corpus, the three active learning methods perform almost identically with little notable difference to distinguish between them. Each method also appreciably outperforms random sampling. Tables 1 and 2 show the test set accuracy and breakeven performance of the active methods after they have asked for just eight labeled instances (so, together with the initial two random instances, they have seen ten labeled instances). They demonstrate that the three active methods perform similarly on this

7. For SVM and transductive SVM learning we used SVMlight Joachims (1999a).

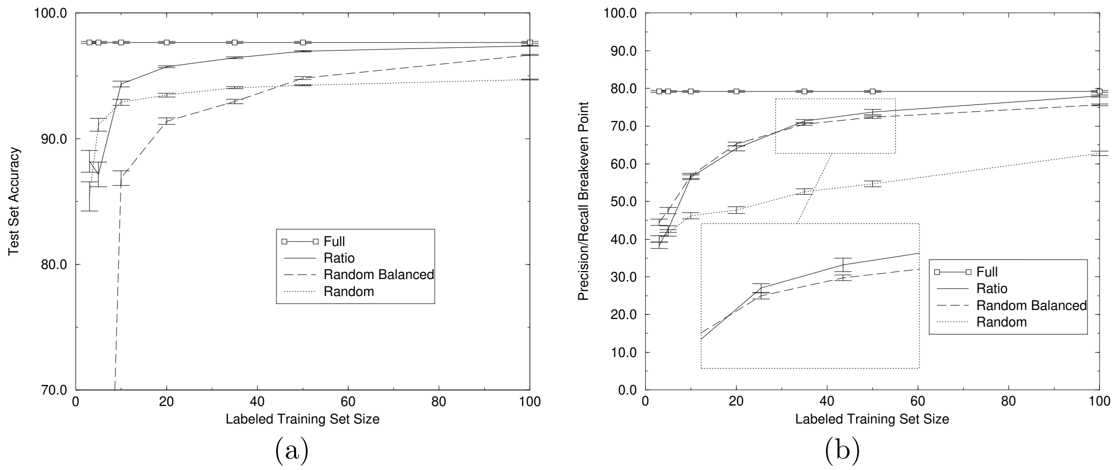
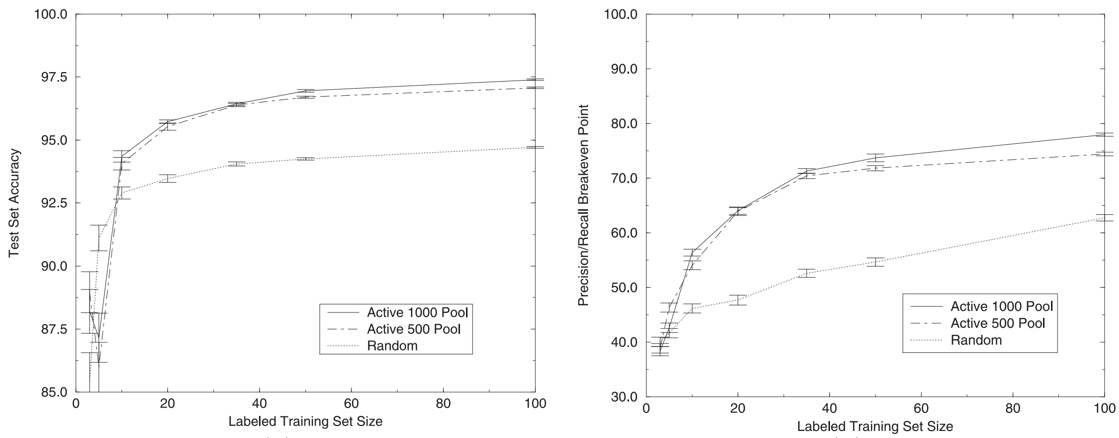


Figure 6: (a) Average test set accuracy over the ten most frequently occurring topics when using a pool size of 1000. (b) Average test set precision/recall breakeven point over the ten most frequently occurring topics when using a pool size of 1000.

Reuters data set after eight queries, with the MaxMin and Ratio showing a very slight edge in performance. The last columns in each table are of more interest. They show approximately how many instances would be needed if we were to use Random to achieve the same level of performance as the Ratio active learning method. In this instance, passive learning on average requires over six times as much data to achieve comparable levels of performance as the active learning methods. The tables indicate that active learning provides more benefit with the infrequent classes, particularly when measuring performance by the precision/recall breakeven point. This last observation has also been noted before in previous empirical tests (McCallum and Nigam, 1998).

We noticed that approximately half of the queries that the active learning methods asked tended to turn out to be positively labeled, regardless of the true overall proportion of positive instances in the domain. We investigated whether the gains that the active learning methods had over regular Random sampling were due to this biased sampling. We created a new querying method called BalancedRandom which would randomly sample an equal number of positive and negative instances from the pool. Obviously in practice the ability to randomly sample an equal number of positive and negative instances without having to label an entire pool of instances first may or may not be reasonable depending upon the domain in question. Figures 6a and 6b show the average accuracy and breakeven point of the BalancedRandom method compared with the Ratio active method and regular Random method on the Reuters dataset with a pool of 1000 unlabled instances. The Ratio and Random curves are the same as those shown in Figures 5a and 5b. The MaxMin and Simple curves are omitted to ease legibility. The BalancedRandom method has a much better precision/recall breakeven performance than the regular Random method, although it is still matched and then outperformed by the active method. For classification accuracy, the BalancedRandom method initially has extremely poor performance (less than 50% which is



(a) (b)

Figure 7: (a) Average test set accuracy over the ten most frequently occurring topics when using a pool sizes of 500 and 1000. (b) Average breakeven point over the ten most frequently occurring topics when using a pool sizes of 500 and 1000.

even worse than pure random guessing) and is always consistently and significantly outperformed by the active method. This indicates that the performance gains of the active methods are not merely due to their ability to bias the class of the instances they queries. The active methods are choosing special targeted instances and approximately half of these instances happen to have positive labels.

Figures 7a and 7b show the average accuracy and breakeven point of the Ratio method with two different pool sizes. Clearly the Random sampling method's performance will not be affected by the pool size. However, the graphs indicate that increasing the pool of unlabeled data will improve both the accuracy and breakeven performance of active learning. This is quite intuitive since a good active method should be able to take advantage of a larger pool of potential queries and ask more targeted questions.

We also investigated active learning in a transductive setting. Here we queried the points as usual except now each method (Simple and Random) returned a transductive SVM trained on both the labeled and remaining unlabeled data in the pool. As described by Joachims (1998) the breakeven point for a TSVM was computed by gradually altering the number of unlabeled instances that we wished the TSVM to label as positive. This invovles re-learning the TSVM multiple times and was computationally intensive. Since our setting was transduction, the performance of each classifier was measured on the pool of data rather than a separate test set. This reflects the relevance feedback transductive inference example presented in the introduction.

Figure 8 shows that using a TSVM provides a slight advantage over a regular SVM in both querying methods (Random and Simple) when comparing breakeven points. However, the graph also shows that active learning provides notably more benefit than transduction—indeed using a TSVM with a Random querying method needs over 100 queries to achieve

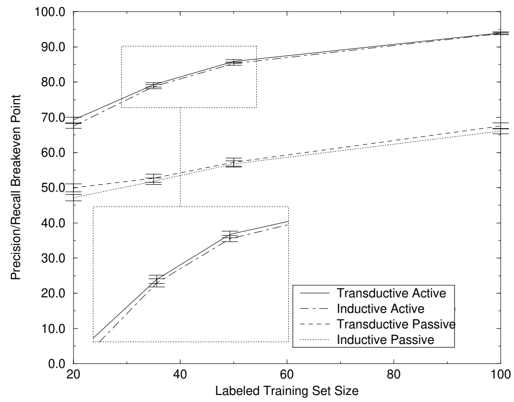


Figure 8: Average pool set precision/recall breakeven point over the ten most frequently occurring topics when using a pool size of 1000.

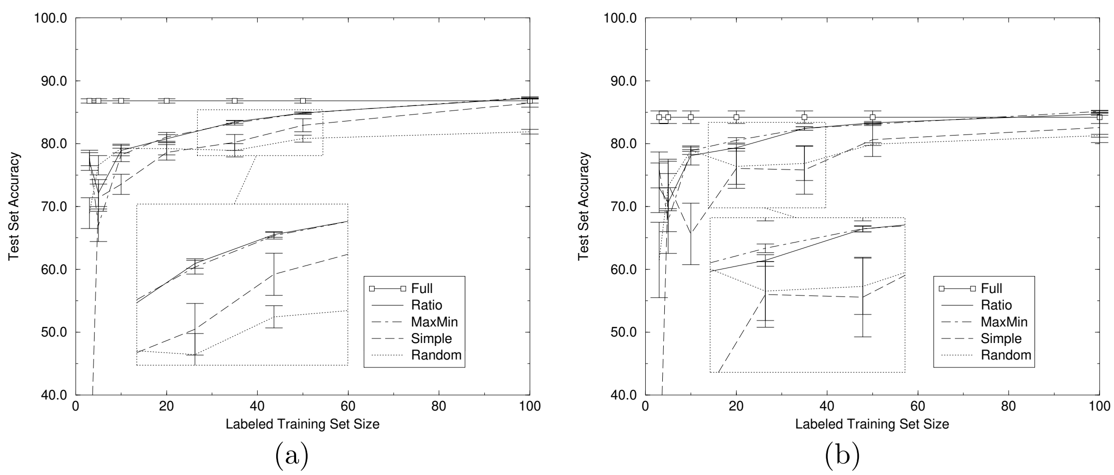


Figure 9: (a) Average test set accuracy over the five `comp.*` topics when using a pool size of 500. (b) Average test set accuracy for `comp.sys.ibm.pc.hardware` with a 500 pool size.

the same breakeven performance as a regular SVM with a Simple method that has only seen 20 labeled instances.

5.2 Newsgroups Experiments

Our second data collection was K. Lang's Newsgroups collection Lang (1995). We used the five `comp.*` groups, discarding the Usenet headers and subject lines. We processed the text documents exactly as before, resulting in vectors of about 10000 dimensions.

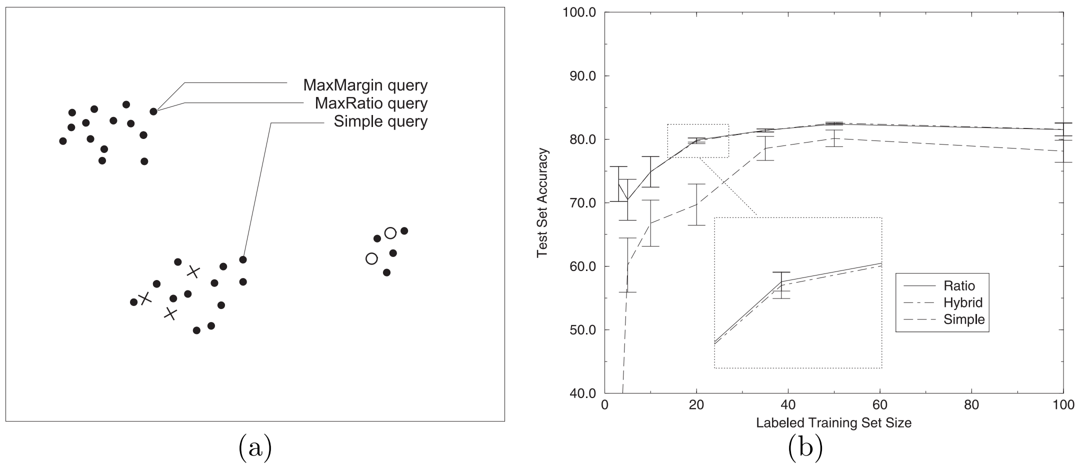


Figure 10: (a) A simple example of querying unlabeled clusters. (b) Macro-average test set accuracy for `comp.os.ms-windows.misc` and `comp.sys.ibm.pc.hardware` where Hybrid uses the Ratio method for the first ten queries and Simple for the rest.

We placed half of the 5000 documents aside to use as an independent test set, and repeatedly, randomly chose a pool of 500 documents from the remaining instances. We performed twenty runs for each of the five topics and averaged the results. We used test set accuracy to measure performance. Figure 9a contains the learning curve (averaged over all of the results for the five `comp.`$*$ topics) for the three active learning methods and Random sampling. Again, the horizontal line indicates the performance of an SVM that has been trained on the entire pool. There is no appreciable difference between the MaxMin and Ratio methods but, in two of the five newsgroups (`comp.sys.ibm.pc.hardware` and `comp.os.ms-windows.misc`) the Simple active learning method performs notably worse than the MaxMin and Ratio methods. Figure 9b shows the average learning curve for the `comp.sys.ibm.pc.hardware` topic. In around ten to fifteen per cent of the runs for both of the two newsgroups the Simple method was misled and performed extremely poorly (for instance, achieving only 25% accuracy even with fifty training instances, which is worse than just randomly guessing a label!). This indicates that the Simple querying method may be more unstable than the other two methods.

One reason for this could be that the Simple method tends not to explore the feature space as aggressively as the other active methods, and can end up ignoring entire clusters of unlabeled instances. In Figure 10a, the Simple method takes several queries before it even considers an instance in the unlabeled cluster while both the MaxMin and Ratio query a point in the unlabeled cluster immediately.

While MaxMin and Ratio appear more stable they are much more computationally intensive. With a large pool of s instances, they require about $2s$ SVMs to be learned for each query. Most of the computational cost is incurred when the number of queries that have already been asked is large. The reason is that the cost of training an SVM grows polynomially with the size of the labeled training set and so now training each SVM is costly (taking

Query	Simple	MaxMin	Ratio	Hybrid
1	0.008	3.7	3.7	3.7
5	0.018	4.1	5.2	5.2
10	0.025	12.5	8.5	8.5
20	0.045	13.6	19.9	0.045
30	0.068	22.5	23.9	0.073
50	0.110	23.2	23.3	0.115
100	0.188	42.8	43.2	0.2

Table 3: Typical run times in seconds for the Active methods on the Newsgroups dataset

over 20 seconds to generate the 50th query on a Sun Ultra 60 450Mhz workstation with a pool of 500 documents). However, when the quantity of labeled data is small, even with a large pool size, MaxMin and Ratio are fairly fast (taking a few seconds per query) since now training each SVM is fairly cheap. Interestingly, it is in the first ten queries that the Simple seems to suffer the most through its lack of aggressive exploration. This motivates a Hybrid method. We can use MaxMin or Ratio for the first few queries and then use the Simple method for the rest. Experiments with the Hybrid method show that it maintains the stability of the MaxMin and Ratio methods while allowing the scalability of the Simple method. Figure 10b compares the Hybrid method with the Ratio and Simple methods on the two newsgroups for which the Simple method performed poorly. The test set accuracy of the Hybrid method is virtually identical to that of the Ratio method while the Hybrid method's run time was about the same as the Simple method, as indicated by Table 3.

6. Related Work

There have been several studies of active learning for classification. The Query by Committee algorithm (Seung et al., 1992, Freund et al., 1997) uses a prior distribution over hypotheses. This general algorithm has been applied in domains and with classifiers for which specifying and sampling from a prior distribution is natural. They have been used with probabilistic models (Dagan and Engelson, 1995) and specifically with the Naive Bayes model for text classification in a Bayesian learning setting (McCallum and Nigam, 1998). The Naive Bayes classifier provides an interpretable model and principled ways to incorporate prior knowledge and data with missing values. However, it typically does not perform as well as discriminative methods such as SVMs, particularly in the text classification domain (Joachims, 1998, Dumais et al., 1998).

We re-created McCallum and Nigam's (1998) experimental setup on the Reuters-21578 corpus and compared the reported results from their algorithm (which we shall call the *MN*-algorithm hereafter) with ours. In line with their experimental setup, queries were asked five at a time, and this was achieved by picking the five instances closest to the current hyperplane. Figure 11a compares McCallum and Nigam's reported results with ours. The graph indicates that the Active SVM performance is significantly better than that of the *MN*-algorithm.

An alternative committee approach to query by committee was explored by Liere and Tadepalli (1997, 2000). Although their algorithm (*LT*-algorithm hereafter) lacks the the-

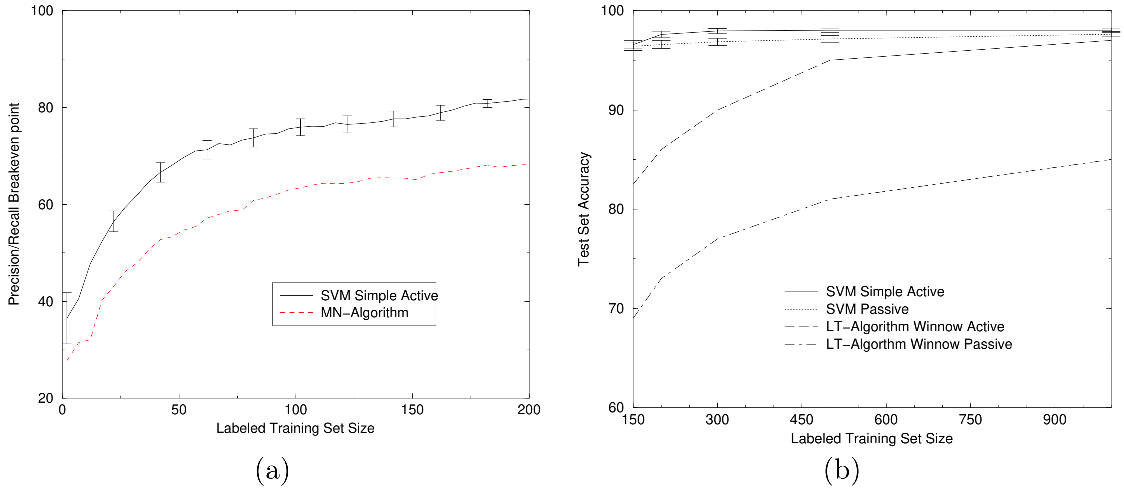


Figure 11: (a) Average breakeven point performance over the Corn, Trade and Acq Reuters-21578 categories. (b) Average test set accuracy over the top ten Reuters-21578 categories.

oretical justifications of the Query by Committee algorithm, they successfully used their committee based active learning method with Winnow classifiers in the text categorization domain. Figure 11b was produced by emulating their experimental setup on the Reuters-21578 data set and it compares their reported results with ours. Their algorithm does not require a positive and negative instance to seed their classifier. Rather than seeding our Active SVM with a positive and negative instance (which would give the Active SVM an unfair advantage) the Active SVM randomly sampled 150 documents for its first 150 queries. This process virtually guaranteed that the training set contained at least one positive instance. The Active SVM then proceeded to query instances actively using the Simple method. Despite the very naive initialization policy for the Active SVM, the graph shows that the Active SVM accuracy is significantly better than that of the LT-algorithm.

Lewis and Gale (1994) introduced uncertainty sampling and applied it to a text domain using logistic regression and, in a companion paper, using decision trees (Lewis and Catlett, 1994). The Simple querying method for SVM active learning is essentially the same as their uncertainty sampling method (choose the instance that our current classifier is most uncertain about), however they provided substantially less justification as to why the algorithm should be effective. They also noted that the performance of the uncertainty sampling method can be variable, performing quite poorly on occasions.

Two other studies (Campbell et al., 2000, Schohn and Cohn, 2000) independently developed our Simple method for active learning with support vector machines and provided different formal analyses. Campbell, Cristianini and Smola extend their analysis for the Simple method to cover the use of soft margin SVMs (Cortes and Vapnik, 1995) with linearly non-separable data. Schohn and Cohn note interesting behaviors of the active learning curves in the presence of outliers.

7. Conclusions and Future Work

We have introduced a new algorithm for performing active learning with SVMs. By taking advantage of the duality between parameter space and feature space, we arrived at three algorithms that attempt to reduce version space as much as possible at each query. We have shown empirically that these techniques can provide considerable gains in both the inductive and transductive settings—in some cases shrinking the need for labeled instances by over an order of magnitude, and in almost all cases reaching the performance achievable on the entire pool having seen only a fraction of the data. Furthermore, larger pools of unlabeled data improve the quality of the resulting classifier.

Of the three main methods presented, the Simple method is computationally the fastest. However, the Simple method seems to be a rougher and more unstable approximation, as we witnessed when it performed poorly on two of the five Newsgroup topics. If asking each query is expensive relative to computing time then using either the MaxMin or Ratio may be preferable. However, if the cost of asking each query is relatively cheap and more emphasis is placed upon fast feedback then the Simple method may be more suitable. In either case, we have shown that the use of these methods for learning can substantially outperform standard passive learning. Furthermore, experiments with the Hybrid method indicate that it is possible to combine the benefits of the Ratio and Simple methods.

The work presented here leads us to many directions of interest. Several studies have noted that gains in computational speed can be obtained at the expense of generalization performance by querying multiple instances at a time (Lewis and Gale, 1994, McCallum and Nigam, 1998). Viewing SVMs in terms of the version space gives an insight as to where the approximations are being made, and this may provide a guide as to which multiple instances are better to query. For instance, it is suboptimal to query two instances whose version space hyperplanes are fairly parallel to each other. So, with the Simple method, instead of blindly choosing to query the two instances that are the closest to the current SVM, it may be better to query two instances that are close to the current SVM and whose hyperplanes in the version space are fairly perpendicular. Similar tradeoffs can be made for the Ratio and MaxMin methods.

Bayes Point Machines (Herbrich et al., 2001) approximately find the center of mass of the version space. Using the Simple method with this point rather than the SVM point in the version space may produce an improvement in performance and stability. The use of Monte Carlo methods to estimate version space areas may also give improvements.

One way of viewing the strategy of always choosing to halve the version space is that we have essentially placed a uniform distribution over the current space of consistent hypotheses and we wish to reduce the expected size of the version space as fast as possible. Rather than maintaining a uniform distribution over consistent hypotheses, it is plausible that the addition of prior knowledge over our hypotheses space may allow us to modify our query algorithm and provided us with an even better strategy. Furthermore, the PAC-Bayesian framework introduced by McAllester (1999) considers the effect of prior knowledge on generalization bounds and this approach may lead to theoretical guarantees for the modified querying algorithms.

Finally, the Ratio and MaxMin methods are computationally expensive since they have to step through each of the unlabeled data instances and learn an SVM for each possible labeling. However, the temporarily modified data sets will only differ by one instance from the original labeled data set and so one can envisage learning an SVM on the original data set and then computing the "incremental" updates to obtain the new SVMs (Cauwenberghs and Poggio, 2001) for each of the possible labelings of each of the unlabeled instances. Thus, one would hopefully obtain a much more efficient implementation of the Ratio and MaxMin methods and hence allow these active learning algorithms to scale up to larger problems.

Acknowledgments

This work was supported by DARPA's *Information Assurance* program under subcontract to SRI International, and by ARO grant DAAH04-96-1-0341 under the MURI program "Integrated Approach to Intelligent Systems".

References

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

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

G Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, volume 13, 2001.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

I. Dagan and S. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. Morgan Kaufmann, 1995.

S.T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*. ACM Press, 1998.

Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting Under Uncertainty. Technical Report MSR-TR-94-07, Microsoft Research, 1994.

R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, pages 245–279, 2001.

E. Horvitz and G. Rutledge. Time dependent utility and action under uncertainty. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1991.

T. Joachims. Text categorization with support vector machines. In *Proceedings of the European Conference on Machine Learning*. Springer-Verlag, 1998.

T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999a.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, 1999b.

K. Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994.

D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag, 1994.

D. McAllester. PAC-Bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. www.cs.cmu.edu/~mccallum/bow, 1996.

A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998.

T. Mitchell. Generalization as search. *Artificial Intelligence*, 28:203–226, 1982.

J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART retrieval system: Experiments in automatic document processing*. Prentice-Hall, 1971.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

Fabrizio Sebastiani. Machine learning in automated text categorisation. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, 2001.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of Computational Learning Theory*, pages 287–294, 1992.

J. Shawe-Taylor and N. Cristianini. Further results on the margin distribution. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 278–285, 1999.

V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag, 1982.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.