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, USA

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 an 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. 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.

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 has been recently introduced (Lewis & Gale, 1994; McCallum & Nigam, 1998). 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 reasonable 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.

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.

Clearly pool-based active learning can be applied to numerous domains — creating automated email filters, interactive searching of an image database, identifying good oil drilling sites. 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. It 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). 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.

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, as well as text classification.

We shall consider SVMs in the binary classification setting. We are given training data $\{\mathbf{x}_1 \ldots \mathbf{x}_n\}$ that are vectors in some space $\mathcal{X} \subseteq \mathbb{R}^d$. We are also given their labels $\{y_1 \ldots y_n\}$ where $y_i \in \{-1, 1\}$. In their simplest form, SVMs are hyperplanes that separate the training data by a maximal margin. 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: $f(\mathbf{x}) = (\sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}))$.

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:

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

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 $\mathcal{X}$. One commonly used kernel is the polynomial kernel $K(\mathbf{u},\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^p$ which induces polynomial boundaries of degree p in the original space $\mathcal{X}$. For the majority of this paper we will assume that the feature vectors are normalized, i.e., $\|\Phi(\mathbf{x})\| = 1$. It is possible to relax this constraint and we will discuss this later in the paper.

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. Recently, *transductive SVMs* (TSVMs) have been used for text classification (Joachims, 1999), 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 3.1 *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:*

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

Note that a version space only exists if the *training* data are linearly separable in feature space. Thus, in this paper, we do require linear separability of the training data in feature space. This restriction is less harsh than it at first may seem since the feature space often has a very high dimension. Nevertheless, requiring linear separability in feature space is a condition we wish to relax in future work.

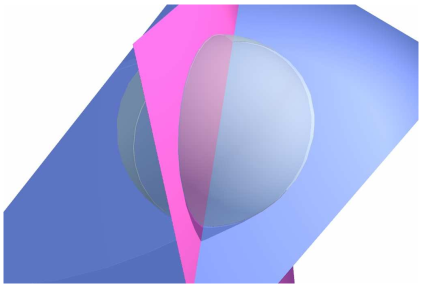
There exists a duality between the feature space $\mathcal{F}$ and the parameter space $\mathcal{W}$ (Vapnik, 1998; Herbrich et al., 1999) which we shall take advantage of in the next section: points in $\mathcal{F}$ correspond to hyperplanes in $\mathcal{W}$ and *vice versa*.

Clearly, 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 feature space restricts the set of separating hyperplanes to ones that classify $\mathbf{x}_i$ correctly. In fact, we can show that the set 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 $y_i\Phi(\mathbf{x}_i)$ as being the normal vector of a hyperplane in $\mathcal{W}$. Thus $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) = \mathbf{w} \cdot y_i\Phi(\mathbf{x}_i) > 0$ defines a half space in $\mathcal{W}$. Furthermore $\mathbf{w} \cdot y_i\Phi(\mathbf{x}_i) = 0$ defines a hyperplane in $\mathcal{W}$ that acts as one of the boundaries to version space $\mathcal{V}$. Notice that version space is a connected region on the surface of a hypersphere in parameter space. See Figure 1(a) for an example.

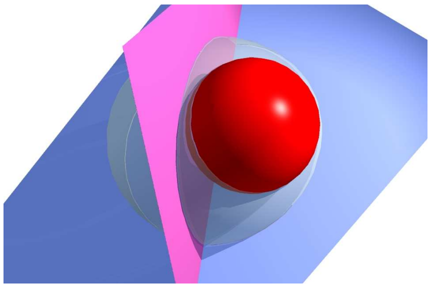
SVMs find the hyperplane that maximizes the margin in feature space $\mathcal{F}$. One way to pose this is as follows:

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

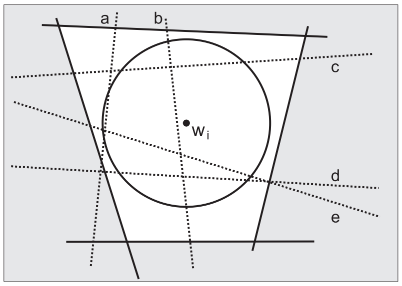
By having the conditions $\|\mathbf{w}\| = 1$ and $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) > 0$ we cause the solution to lie in version space. Now, we can view the above problem as finding the point $\mathbf{w}$ in version space that maximizes the distance $\min_i \{ \mathbf{w} \cdot y_i\Phi(\mathbf{x}_i) \}$. From the duality between feature and parameter space, and since $\|\Phi(\mathbf{x}_i)\| = 1$, each $y_i\Phi(\mathbf{x}_i)$ is a unit normal vector of a hyperplane in parameter space and each of these hyperplanes delimit the version space. Thus we want to find the point in 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 version space and whose surface does not intersect with the hyperplanes corresponding to the labeled instances, as in Figure 1(b). It can be easily shown that the hyperplanes that are touched by the maximal radius hypersphere correspond to the support vectors and that the radius of the hypersphere is the margin of the SVM.



(a)



(b)



(c)

Figure 1. (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 version space is the surface segment of the hypersphere closest to the camera. (b) An SVM classifier in version space. The dark embedded sphere is the largest radius sphere whose center lies in version space and whose surface does not intersect with the hyperplanes. The center of the embedded sphere corresponds to the SVM, its radius is the margin of the SVM in $\mathcal{F}$ and the training points corresponding to the hyperplanes that it touches are the support vectors. (c) Simple Margin Method.

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 a set 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. As in 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 need two more definitions before we can proceed:

Definition 4.1 *$Area(\mathcal{V})$ is the surface area that the version space $\mathcal{V}$ occupies on the hypersphere $\|\mathbf{w}\| = 1$.*

Definition 4.2 *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 can now appeal to the following lemma to motivate which instances to query:

Lemma 4.3 *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,*

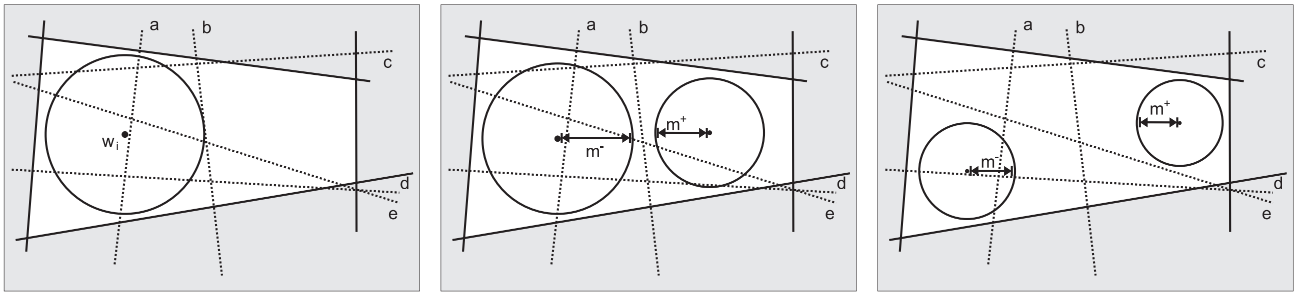
$$\forall i \in \mathbb{N}^+ \sup_{P \in \mathcal{P}} E_P[Area(\mathcal{V}_i^*)] \leq \sup_{P \in \mathcal{P}} E_P[Area(\mathcal{V}_i)],$$

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

The proof is straightforward and appears in the longer version of this paper. This lemma says that, for any given number of queries, ℓ^* minimizes the maximum expected size of the version space, where the maximum is taken over all conditional distributions of y given $\mathbf{x}$. Suppose $w^* \in \mathcal{W}$ 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 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 w^* can lie. Hence, the SVM that we learn from our limited number of queries will lie close to 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).

Hence, we would like to query instances that split the current version space into two equal parts as much as possi-



(a) (b) (c)

Figure 2. (a) Simple Margin will query $\mathbf{a}$. (b) MaxMin Margin will query $\mathbf{b}$. The two SVMs with margins m^- and m^+ for $\mathbf{b}$ are shown. (c) MaxRatio Margin will query $\mathbf{e}$. The two SVMs with margins m^- and m^+ for $\mathbf{e}$ are shown.

ble. 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 version space, and the more it bisects 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 1(c) presents an illustration. In this stylized picture we have flattened out the surface of the unit weight vector hypersphere that appears in 1(a). 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 version space. Note that the edges of the circle do not touch the solid lines — just as the dark sphere in 1(b) does not meet the hyperplanes on the surface of the larger hypersphere (they meet somewhere under the surface). Instance $\mathbf{b}$ is the 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 version space being fairly symmetric and $\mathbf{w}_i$ being centrally placed. It has been demonstrated, both in theory and practice, that these assumptions can fail significantly (Herbrich et al., 1999). 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 somewhat overcome these problems. 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 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 and equal split of the version space, we wish $\mathit{Area}(\mathcal{V}^-)$ and $\mathit{Area}(\mathcal{V}^+)$ to be similar. Now, consider the quantity $\min(\mathit{Area}(\mathcal{V}^-), \mathit{Area}(\mathcal{V}^+))$. It will be small if $\mathit{Area}(\mathcal{V}^-)$ and $\mathit{Area}(\mathcal{V}^+)$ 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 2(a) and 2(b) show an example comparing the Simple Margin and MaxMin Margin methods.

• **MaxRatio 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 take into account the fact that the current version space

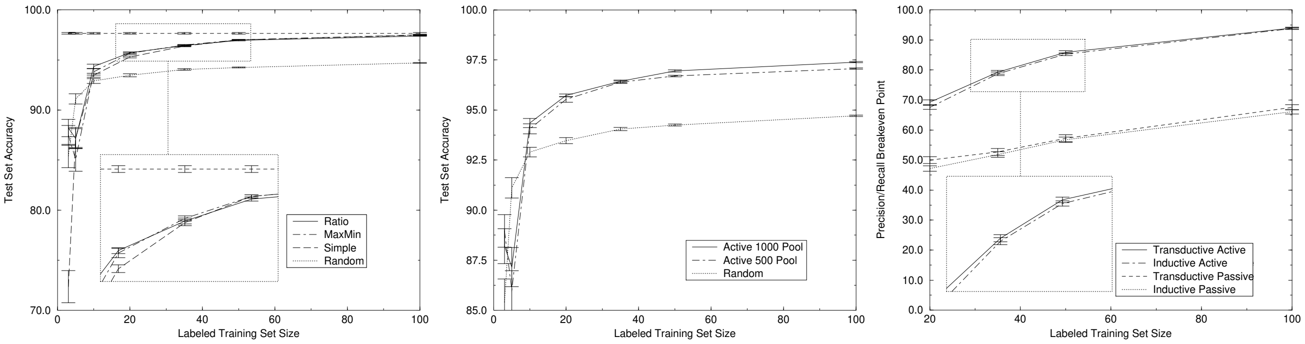


Figure 3. (a) Average test set accuracy over the ten most frequently occurring topics when using a pool size of 1000. (b) Average test set accuracy over the ten most frequently occurring topics when using a pool sizes of 500 and 1000. (c) Average pool set precision/recall breakeven point over the ten most frequently occurring topics when using a pool size of 1000.

$\mathcal{V}_i$ 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 2(c)).

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 are normalized. Thus the explanation for the MaxMin and MaxRatio methods still holds even with unnormalized feature vectors. The Simple method can still be used when the feature vector are unnormalized, however the motivating explanation no longer holds since the SVM can no longer be viewed as the center of the largest allowable sphere.

5. Experiments

5.1 Reuters Data Collection Experiments

The Reuters-21578 data set[1] is a commonly used collection of newswire stories categorized into hand labeled topics, some of which overlap. We used the 12902 articles from the "ModApte" split of the data and 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 unit vector.[2] A stop list of common words was used and words occurring in less than three documents were also ignored. Using this representation, the document vectors had around 10000 dimensions. Our test set consisted of 3299 documents.

For each of the ten topics we performed the following. 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 SVM with a polynomial kernel of degree one[3] 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 MaxRatio 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).

Figure 3(a) present the average test set accuracy over the ten topics as we vary the number of queries permitted. The precision/recall breakeven point graph is virtually identical and is omitted. The dashed 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 virtually identically with no notable difference to distinguish between them. Each method also appreciably out-

[1]Obtained from www.research.att.com/˜lewis.

[2]We used Rainbow (www.cs.cmu.edu/˜mccallum/bow) for text processing.

[3]For SVM and transductive SVM learning we used T. Joachims' SVMlight: www-ai.informatik.uni-dortmund.de/thorsten/svm_light.html.

Table 1. 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.

Topic	Simple	MaxMin	MaxRatio	Equivalent Random size
Earn	86.05	89.03	88.95	12
Acq	54.14	56.43	57.25	12
Money-fx	35.62	38.83	38.27	52
Grain	50.25	58.19	60.34	51
Crude	58.22	55.52	58.41	55
Trade	50.71	48.78	50.57	85
Interest	40.61	45.95	43.71	60
Ship	53.93	52.73	53.75	> 100
Wheat	64.13	66.71	66.57	> 100
Corn	49.52	48.04	46.25	> 100

performs random sampling. Table 1 shows the 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). The last column shows approximately how many instances would be needed if we were to use Random to achieve the same level of performance as the MaxRatio 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 table for test set accuracy is very similar and is omitted. Table 1 indicates that active learning provides more benefit with the infrequent classes. This last observation has also been noted before in previous studies (McCallum & Nigam, 1998).

Figure 3(b) shows the average accuracy of the MaxRatio method with two different pool sizes. The breakeven graph is similar. 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. Since this was transduction the performance of each classifier was measured on the pool of data rather than a separate test set. Figure 3(c) shows that using a TSVM provides a slight advantage over a regular SVM in both querying methods 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 the same performance as a regular SVM with a Simple method that has only seen 20 labeled instances. Interestingly, we found that the TSVM *misclassification* rates were notably worse than those of regular SVMs. The breakeven point is a one number summary of the precision/recall (P/R) curve. Typically, each TSVM P/R curve was very erratic, only peaking around the breakeven point . So, although we happened to get better breakeven performance, most of a typical TSVM's P/R curve was actually worse than that of the corresponding SVM. It could also be because the TSVM package we used was an approximation to the (infeasible) global solution.

5.2 Newsgroups Data Collection Experiments

Our second data collection was Ken Lang's Newsgroups collection[4]. 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 around 10000 dimensions.

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 4(a) 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 dashed 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 MaxRatio 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 MaxRatio methods. Figure 4(b) 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 random guessing!). 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 5 the Simple method takes several queries before it even considers an instance in the unlabeled cluster while both the MaxMin and MaxRatio query a point in the unlabeled cluster immediately.

While MaxMin and MaxRatio appear more stable they are much more computationally intensive. With a large pool of s instances, they require around $2s$ SVMs to be learned

[4]Obtained from www.cs.cmu.edu/~textlearning.

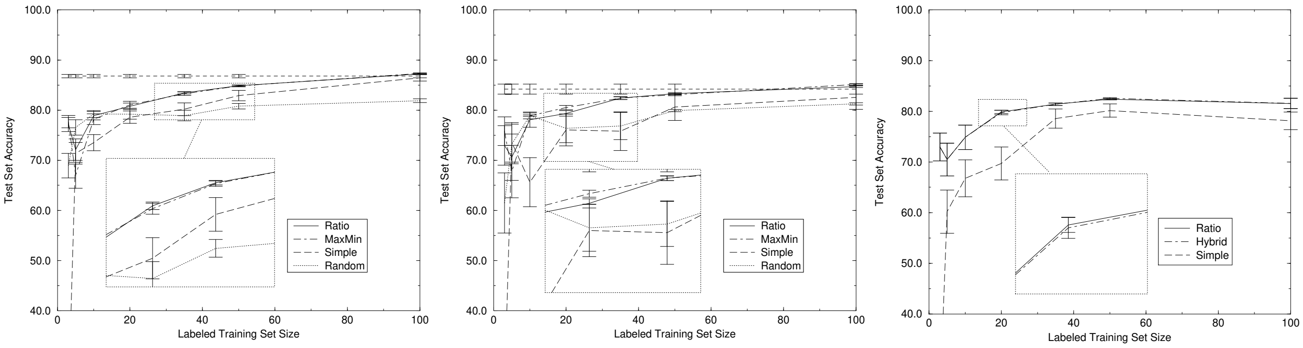


Figure 4. (a) Average test set accuracy over the five `comp.*` topics when using a pool size of 500. (b) Average test set accuracy for `comp.sys.ibm.pc.hardware` with a 500 pool size. (c) Macro average test set accuracy for `comp.os.ms-windows.misc` and `comp.sys.ibm.pc.hardware` where Hybrid uses the MaxRatio method for the first ten queries and Simple for the rest.

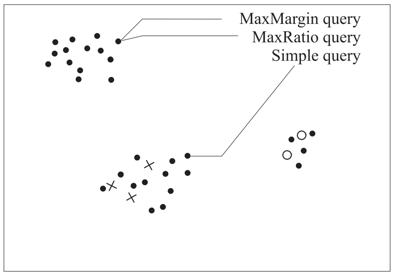


Figure 5. Simple example of querying unlabeled clusters.

for each query. Most of the computational cost is incurred when the number of queries that have already been asked is large. This is because that now training each SVM is costly (taking around two minutes to generate the 50th query on a Sun workstation with a pool of 1000 documents). However, when the number of already-asked queries is small, even with a large pool size, MaxMin and MaxRatio are fairly fast (taking a few seconds per query) since now training each SVM is fairly cheap. Interestingly, is it in the first few 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 MaxRatio for the first few queries and then use the Simple method for the rest. Preliminary experiments with the Hybrid method show that it maintains the stability of the MaxMin and MaxRatio methods while allowing the scalability of the Simple method. See Figure 4(c).

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 & Engelson, 1995) and specifically with the Naive Bayes model for text classification in a Bayesian learning setting (McCallum & 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). Although a direct comparison has not been explicitly made here, the results of the active SVMs presented in this paper are generally significantly better than those of the active Naive Bayes approach investigated by McCallum and Nigam (1998).

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 & 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.

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 only seen a fraction of the data. Furthermore, larger pools of unlabeled data improve the quality of the resulting classifier.

Of the three methods presented the Simple method is computationally the fastest. However, the Simple method would seem 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 (such as in the oil drilling example) then using either the MaxMin or MaxRatio 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 MaxRatio and Simple methods.

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 & Gale, 1994; McCallum & 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 version space are fairly perpendicular. Similar tradeoffs can be made for the MaxRatio and MaxMin methods.

Clearly the three active learning methods described in this paper can be applied to other domains. However, text classification is a particularly suitable domain for our algorithms as they stand at the moment. Sets of documents are almost always linearly separable which is an important consideration for the time being. As we mentioned before, the notion of a version space only holds when the training data are linearly separable in feature space. If the data are not separable then there is no version space. However, SVMs can still learn with linearly non-separable data via the use of a *soft margin* (Cortes & Vapnik, 1995). One can envisage using the soft margin value (or a similar quantity) to approximate the "size" of the "version space" when dealing with an hypothesis class of Soft Margin SVMs. Indeed, preliminary experiments for algorithms that work along these lines have produced promising results. We would like to extend our analysis for this scenario as well as apply our active learning algorithms to a wider class of domains.

Acknowledgements

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

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

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 1–25.

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

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

Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1997). Selective sampling using the Query by Committee algorithm. *Machine Learning*, *28*, 133–168.

Herbrich, R., Graepel, T., & Campbell, C. (1999). Bayes point machines: Estimating the bayes point in kernel space. *International Joint Conference on Artificial Intelligence Workshop on Support Vector Machines* (pp. 23–27).

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

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

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

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

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

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

Rocchio, J. (1971). Relevance feedback in information retrieval. *The SMART retrival system: Experiments in automatic document processing*. Prentice-Hall.

Seung, H., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Proceedings of the Fifth Workshop on Computational Learning Theory* (pp. 287–294). Morgan Kaufmann.

Vapnik, V. (1982). *Estimation of dependences based on empirical data*. Springer Verlag.

Vapnik, V. (1998). *Statistical learning theory*. Wiley.