Toward Optimal Active Learning through Sampling Estimation of Error Reduction

Nicholas Roy NICHOLAS.ROY@RI.CMU.EDU
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Andrew McCallum MCCALLUM@WHIZBANG.COM
WhizBang! Labs - Research, 4616 Henry Street, Pittsburgh, PA 15213 USA

Abstract

This paper presents an active learning method that directly optimizes expected future error. This is in contrast to many other popular techniques that instead aim to reduce version space size. These other methods are popular because for many learning models, closed form calculation of the expected future error is intractable. Our approach is made feasible by taking a sampling approach to estimating the expected reduction in error due to the labeling of a query. In experimental results on two real-world data sets we reach high accuracy very quickly, sometimes with four times fewer labeled examples than competing methods.

1. Introduction

Traditional supervised learning methods set their parameters using whatever training data is given to them. By contrast, *active learning* is a framework in which the learner has the freedom to select which data points are added to its training set. An active learner may begin with a very small number of labeled examples, carefully select a few additional examples for which it requests labels, learn from the result of that request, and then using its newly-gained knowledge, carefully choose which examples to request next. In this way the active learner aims to reach high performance using as few labeled examples as possible. Thus active learning can be invaluable in the common case in which there are limited resources for labeling data, and obtaining these labels is time-consuming or difficult.

Cohn et al. (1996) describe a statistically optimal solution to this problem. Their method selects the training example that, once labeled and added to the training data, is expected to result in the lowest error on future test examples. They develop their method for two simple regression problems in which this question can be answered in closed form. Unfortunately there are many tasks and models for which the optimal selection cannot efficiently be found in closed form.

Other, more widely used active learning methods attain practicality by optimizing a different, non-optimal criterion. For example, *uncertainty sampling* (Lewis & Gale, 1994) selects the example on which the current learner has lowest certainty; *Query-by-Committee* (Seung et al., 1992; Freund et al., 1997) selects examples that reduce the size of the version space (Mitchell, 1982) (the size of the subset of parameter space that correctly classifies the labeled examples). Tong and Koller's Support Vector Machine method (2000a) is also based on reducing version space size. None of these methods directly optimize the metric by which the learner will ultimately be evaluated—the learner's expected error on future test examples. Uncertainty sampling often fails by selecting examples that are outliers—they have high uncertainty, but getting their labels doesn't help the learner on the bulk of the test distribution. Version-space reducing methods, such as Query-by-Committee often fail by spending effort eliminating areas of parameter space that have no effect on the error rate. Thus these methods also are not immune to selecting outliers; see McCallum and Nigam (1998b) for examples.

This paper presents an active learning method that combines the best of both worlds. Our method selects the next example according to the optimal criterion (reduced error rate on future test examples), but solves the practicality problem by using sampling estimation. We describe our method in the framework of document classification with pool-based sampling, but it would also apply to other forms of classification or regression, and to generative sampling. We describe an implementation in terms of naive Bayes, but the same technique could apply to any learning method in which incremental training is efficient—for example support vector machines (SVMs) (Cauwenberghs & Poggio, 2000).

Our method estimates future error rate either by log-loss, using the entropy of the posterior class distribution on a sample of the unlabeled examples, or by 0-1 loss, using the posterior probability of the most probable class for the sampled unlabeled examples. At each round of active learning, we select an example for labeling by sampling from the unlabeled examples, adding it to the training set with a sample of its possible labels, and estimating the resulting future error rate as just described. This seemingly daunting sampling and re-training can be made efficient through a number of rearrangements of computation, careful sampling choices, and efficient incremental training procedures for the underlying learner.

We show experimental results on two real-world document classification tasks, where, in comparison with density-weighted Query-by-Committee we reach 85% of full performance in one-quarter the number of training examples.

2. Optimal Active Learning and Sampling Estimation

The optimal active learner is one that asks for labels on the examples that, once incorporated into training, will result in the lowest expected error on the test set.

Let $P(y|x)$ be an unknown conditional distribution over inputs, x, and output classes, $y \in \{y_1, y_2, \ldots y_n\}$, and let $P(x)$ be the marginal "input" distribution. The learner is given a labeled training set, $\mathcal{D}$, consisting of IID input/output pairs drawn from $P(x)P(y|x)$, and estimates a classification function that, given an input x, produces an estimated output distribution $\hat{P}_{\mathcal{D}}(y|x)$. We can then write the expected error of the learner as follows:

$$E_{\hat{P}_{\mathcal{D}}} = \int_x L(P(y|x), \hat{P}_{\mathcal{D}}(y|x))\ P(x), \qquad (1)$$

where L is some loss function that measures the degree of our disappointment in any differences between the true distribution, $P(y|x)$ and the learner's prediction, $\hat{P}_{\mathcal{D}}(y|x)$. Two common loss functions are:
log loss: $L = \sum_{y \in \mathcal{Y}} P(y|x) \log(\hat{P}_{\mathcal{D}}(y|x))$
and 0/1 loss:
$L = \sum_{y \in \mathcal{Y}} P(y|x)(1 - \delta(y, \arg\max_{y' \in \mathcal{Y}} \hat{P}_{\mathcal{D}}(y'|x))$.

First-order Markov active learning thus aims to select a query, $x^\star$, such that when the query is given label $y^\star$ and added to the training set, the learner trained on the resulting set $(\mathcal{D} + (x^\star, y^\star))$ has lower error than any other x,

$$\forall (x,y)\ E_{\hat{P}_{\mathcal{D}+(x^\star,y^\star)}} < E_{\hat{P}_{\mathcal{D}+(x,y)}}. \qquad (2)$$

We concern ourselves here with *pool-based active learning*, in which the learner has available a large pool, $\mathcal{P}$, of unlabeled examples sampled from $P(x)$, and the queries may be chosen only from this pool. The pool thus not only provides us with a finite set of queries, but also an estimate of $P(x)$.

This paper takes a sampling approach to error estimation and the choice of query. Rather than estimating expected error over the full distribution, $P(x)$, we measure it over the sample in the pool. Furthermore, the true output distribution $P(y|x)$ is unknown for each sample x, so we estimate it using the current learner.[1] (For log loss this results in estimating the error by the entropy of the learner's posterior distribution).

Writing the labeled documents $\mathcal{D} + (x^\star, y^\star)$ as $\mathcal{D}^\star$, for log loss we have

$$\tilde{E}_{\hat{P}_{\mathcal{D}^\star}} = \frac{1}{|\mathcal{P}|} \sum_{x \in \mathcal{P}} \sum_{y \in \mathcal{Y}} \hat{P}_{\mathcal{D}^\star}(y|x) \log(\hat{P}_{\mathcal{D}^\star}(y|x)), \qquad (3)$$

and for 0/1 loss

$$\tilde{E}_{\hat{P}_{\mathcal{D}^\star}} = \frac{1}{|\mathcal{P}|} \sum_{x \in \mathcal{P}} \left(1 - \max_{y \in \mathcal{Y}} \hat{P}_{\mathcal{D}^\star}(y|x)\right). \qquad (4)$$

Of course, before we make the query, the true label for $x^\star$ is also unknown. Again, the current learned classifier gives us an estimate of the distribution from which the $x^\star$'s true label would be chosen, $\hat{P}_{\mathcal{D}}(y|x^\star)$, and we can use this in an expectation calculation by calculating the estimated error for each possible label, $y \in \{y_1, y_2, \ldots y_n\}$, and taking an average weighted by the current classifier's posterior, $\hat{P}_{\mathcal{D}}(y|x^\star)$ of $\tilde{E}_{\hat{P}_{\mathcal{D}^\star}}$.

In the above formulation, we are using the current learner to estimate the true label probabilities, which may seem counter-intuitive. Using these loss functions will cause the learner to select those examples which maximizes the sharpness of learner's posterior belief about the unlabeled examples. An example will be selected if it dramatically reinforces the learner's existing belief over unlabeled examples for which it is currently unsure. In practice, selecting these instances for labeling is reasonable because the most useful (or informative) labelings are usually consistent with the learner's prior belief over the majority (but not all) of unlabeled examples.

Our algorithm thus consists of the following steps:

```
1. train a classifier using the current labeled examples
   (a) consider each unlabeled example, $x$, in the pool as a candidate for the next labeling request
       i. consider each possible label, $y$, for $x$, and add the pair $(x, y)$ to the training set
       ii. re-train the classifier with the enlarged training set, $\mathcal{D} + (x, y)$
       iii. estimate the resulting expected loss as in equation (3) or equation (4).
   (b) Assign to $x$ the average expected losses for each possible labeling, $y$, weighted according to the current classifier's posterior, $\hat{P}_{\mathcal{D}}(y|x)$
2. Select for labeling the unlabeled example $x$ that generated the lowest expected error on all other examples.
```

If implemented naively, the above algorithm would be hopelessly inefficient. However, with some thought and

[1] In order to reduce variance of this estimate we create several training sets by sampling with replacement from the labeled set (bagging), and averaging the resulting posterior class distribution. See section 3.2 for more details.

some rearrangements of computation, there are a number of optimizations and approximations that make this algorithm much more efficient and very tractable:

- Most importantly, many learning algorithms have algorithms for very efficient *incremental* training. That is, the cost of re-training after adding one more example to the training set is far less than re-training as if the entire set were new. For example, in a naive Bayes classifier, only a few event counts need be incremented. SVMs also have efficient re-training procedures (Cauwenberghs & Poggio, 2000).
- Furthermore, many learners have efficient algorithms for incremental *re-classification* of the examples in the pool. In incremental re-classification, the only parts of computation that need to be redone are those that would have changed as a result of the additional training instance. Again, naive Bayes and SVMs are two examples of algorithms that permit this.
- After adding a candidate query to the training set, we do not need to re-estimate the error associated with all other examples in the pool—only those likely to be effected by the inclusion of the candidate in the training set. In many cases this means simply skipping examples not in the "neighborhood" of the candidate or skipping examples without any features that overlap with the features of the candidate. Inverted indices, in which all the examples containing a particular features are listed together, can make this extremely efficient.
- The pool of candidate queries can be reduced by random sub-sampling, or pre-filtering to remove outliers according to some criteria. In fact, any of the suboptimal active learning methods might make good pre-filters.
- The expected error can be estimated using only a subsample of the pool. Especially when the pool is large, there is no need to use all examples—a good estimate may be formed with only few hundred examples.

In the remainder of the paper we describe a naive Bayes implementation of our method, discuss related work, and present experimental results on two real-world data sets showing that our method significantly outperforms methods optimize indirect criteria, such as query uncertainty. We also outline some future work.

3. Naive Bayes Text Classification

Text classification is not only a task of tremendous practical significance, but is also an interesting problem in machine learning because it involves an unusually large number of features, and thus requires estimating an unusually large number of parameters. It is also a domain in which obtaining labeled data is expensive, since the human effort of reading and assigning documents to categories is almost always required. Hence, the large number of parameters often must be estimated from a small amount of labeled data. When little training data is being used to estimate the parameters for a large number of features, it is often best to use a simple learning model. In such cases, there is not enough data to support estimations of feature correlations and other complex interactions. One such classification method that performs surprisingly well given its simplicity is *naive Bayes*. Naive Bayes is not always the best performing classification algorithm for text (Nigam et al., 1999; Joachims, 1998), but it continues to be widely used for the purpose because it is efficient and simple to implement, and even against significantly more complex methods, it rarely trails far behind in accuracy.

This paper's sampling approach to active learning could be applied to several different learners. We apply it here to naive Bayes for the sake of simplicity of explanation and implementation. Experiments with other learners is an item of future work.

Naive Bayes is a Bayesian classifier based on a generative model in which data are produced by first selecting a class, $y \in \mathcal{Y}$, and then generating features of the instance, $x \in \mathcal{X}$, independently given the class. For text classification, the common variant of naive Bayes has unordered word counts for features, and uses a per-class multinomial to generate the words (McCallum & Nigam, 1998a). Let w_t be the tth word in the dictionary of words $\mathcal{V}$, and $\theta = (\theta_{y_j}, \theta_{w_t|y_j})_{y_j \in \mathcal{Y}, w_t \in \mathcal{V}}$ be the parameters of the model, where θ_{y_j} is the prior probability of class y_j (otherwise written $P(y_j|\theta)$), and where $\theta_{w_t|y_j}$ is the probability of generating word w_t from the multinomial associated with class y_j (otherwise written $P(w_t|y_j, \theta)$, and $\sum_t \theta_{w_t|y_j} = 1$ for all y_j.

Thus the probability of generating the ith instance, x_i is

$$P(x_i|\theta) \propto \sum_{j=1}^{|\mathcal{Y}|} P(y_j|\theta) \prod_{k=1}^{|x_i|} P(w_{x_{ik}}|y_j, \theta), \qquad (5)$$

where x_{ik} is the kth word in document x_i. Then, by Bayes rule, the probability that document x_i was generated by class y_j is

$$P(y_j|x_i, \theta) = \frac{P(y_j|\theta) \prod_{k=1}^{|x_i|} P(w_{x_{ik}}|y_j, \theta)}{\sum_{r=1}^{|\mathcal{Y}|} P(y_r, \theta) \prod_{k=1}^{|x_i|} P(w_{x_{ik}}|y_r, \theta)}. \qquad (6)$$

Maximum *a posteriori* parameter estimation is performed by taking ratios of counts:

$$\hat{\theta}_{w_t|y_j} = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} N(w_t, x_i) P(y_j|x_i)}{|\mathcal{V}| + \sum_{s=1}^{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{D}|} N(w_s, x_i) P(y_j|x_i)}, \qquad (7)$$

$$\hat{\theta}_{y_j} = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} P(y_j|x_i)}{|\mathcal{Y} + |\mathcal{D}|}, \qquad (8)$$

where $N(w_t, x_i)$ is the number of times word w_t occurs in document x_i, and $P(y_j|x_i)$ is an indicator variable that is 1 when document x_i has label y_j and 0 otherwise.

3.1 Fast Naive Bayes Updates

Equations (3) and (4) show how the pool of unlabeled documents can be used to estimate the change in classifier error if we label one document. However, in order to choose the best candidate from the pool of unlabeled documents $\mathcal{P}$, we have to train the classifier $|\mathcal{P}|$ times, and each time classify $|\mathcal{P}| - 1$ documents. Performing $\mathcal{O}(|\mathcal{P}|^2)$ classifications for every query can be computationally infeasible for large document pools.

While we cannot reduce the total number of classifications for every query, we can take advantage of certain data structures in a naive Bayes classifier to allow more efficient retraining of the classifier and relabeling of each unlabeled document.

Recall from equation (6) that each class probability for an unlabeled document is a product of the word probabilities for that label. When we compute the class probabilities for each unlabeled document using the new classifier, we make an approximation by only modifying some of the word probabilities in the product of equation (6). By propagating only changes to word probabilities for words in the putatively labeled document, we gain substantial computational savings compared to retraining and reclassifying each document.

Given a classifier learned from training data $\mathcal{D}$, we can add a new document x_p with label y_p to the training data and update the class probabilities (for that class y_p) of each unlabeled document x_i in $\mathcal{P}$ by:

$$P(y_p|x_i, \hat{\theta}') = \frac{P(y_p|x_i, \hat{\theta}) \prod_{w \in x_i \cap x_p} P'(w|y_p, \hat{\theta}')}{\prod_{w \in x_i \cap x_p} P(w|y_p, \hat{\theta})}, \quad (9)$$

where $P(w|y_p, \hat{\theta}')$ are the new word probabilities given $\mathcal{D} + (x_p, y_p)$, and $P(w|y_p, \hat{\theta})$ are the old word probabilities given only $\mathcal{D}$. The denominator divides out the old multinomials from the previous classifier. The product in the right hand side of the numerator multiplies in the new word probabilities that result from adding the putatively labeled document x_p.

The old multinomials that are divided out are the same as in equation (6). The new multinomials for the numerator can be obtained rapidly by incrementally adding to the word counts, (*i.e.* only the first terms of the numerator and denominator need to be added to the pre-existing counts for the rest of the numerator and denominator):

$$\begin{aligned} P(w_t|y_p, \hat{\theta}') \quad &= \quad (10) \\ &\frac{N(w_t, x_p) + 1 + \sum_{i=1}^{|\mathcal{D}|} N(w_t, x_i) P(y_p|x_i)}{|x_p| + |\mathcal{V}| + \sum_{s=1}^{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{D}|} N(w_s, x_i) P(y_p|x_i)}, \end{aligned}$$

where $N(w_t, x_p)$ is the word count for a word w_t in the putatively labeled document x_p. Again, we only do this for the label probabilities of the putative label y_p; all other label probabilities remain unchanged.

3.2 Obtaining Smoother Posteriors for Naive Bayes

Our active learning method relies on obtaining reasonably accurate class posteriors from the classification procedure. It is well-known that naive Bayes, with its violated independence assumption, gives overly sharp posteriors—the probability of the winning class tends to be very close to 1, and the losing classes have probabilities close to 0.

We address this problem with a sampling-based approach to variance reduction, otherwise known as *bagging* (Breiman, 1996). From our original labeled training set of size s, a different training set is created by sampling s times with replacement from the original. The learner then creates a new classifier from this sample, this procedure is repeated m times, and the final class posterior for an instance is taken to be the unweighted average of the class posteriors for each of the classifiers. For each round in which a new query is to be chosen, these training set bags are resampled, and each putatively labeled document is temporarily added to each bag in turn.

In regions of uncertain classification is it often the case that the classifiers from different samples give different answers. Thus, even when the posteriors from any individual classifier are completely extreme, the bagged posterior is more smooth and reflective of the true uncertainty. This approach has been shown not necessarily to reduce overfitting (Domingos, 2000), but it does certainly give better posterior probabilities.

One interesting aspect of this approach is that it can be applied to any classifier—even ones that don't give class posterior probabilities at all, or for which the distribution over classifier parameters is unclear. This "bagging approach" to sampling from the distribution over classifiers has been used in previous work related to QBC (Abe & Mamitsuka, 1998); see the related work section for more details.

4. Related Work

Cohn et al. (1996) propose one of the first statistical analyses of active learning, demonstrating how to construct queries that maximize the error reduction by minimizing the learner's variance. They take advantage of the fact that an unbiased learner that minimizes the expected error given as the expected sum of squared error is equivalent to an unbiased learner that minimizes its variance. Such a learner can then use the estimated distribution of $\hat{y}$ to estimate $\langle \hat{\sigma}_y^2 \rangle$, the expected variance of the learner after querying at $\hat{x}$. However, a closed-form solution for the expected variance of the text classifier is difficult to compute. Furthermore, they construct exactly the query that maximizes this reduction, rather than choosing from a pool of possible queries.

Cohn et al.'s "Constructive Query Generation" approach is contrasted with "Query-Filtering" (or Seung et al. (1992)'s

"Selective Sampling"), in which unlabeled data is presented to the learner from some distribution, and the learner chooses queries from this sample (either as a pool or a stream). From this data-oriented perspective, Lewis and Gale (1994) presented the *uncertainty sampling* algorithm for choosing the example with the greatest uncertainty in predicted label. Freund et al. (1997) showed that *uncertainty sampling* does not converge to the optimal classifier as quickly as the "Query-By-Committee" algorithm (Seung et al., 1992).

In the "Query By Committee" (QBC) approach, the method is to reduce the error of the learner by choosing the instance to be labeled that will minimize the size of the version space (Mitchell, 1982) consistent with the labeled examples. Instead of explicitly determining the size of the version space, predicted labels for each unlabeled example are generated by first drawing hypotheses probabilistically from the version space, according to a distribution over the concepts in the version space. These hypotheses are then used to predict the example label. Examples arrive from a stream, and are labeled whenever the 'committee' of hypotheses disagree on the predicted label. This approach chooses examples that "split the version space into two parts of comparable size" with a degree of probability that guarantees data efficiency that is logarithmic in the desired probability of error.

A number of others have made use of QBC-style algorithms; in particular, Liere and Tadepalli (1997) use committees of Winnow learners for text classification, and Argamon-Engelson and Dagan (1999) use QBC for natural language processing. Our algorithm differs from theirs in that we are estimating the error reduction, whereas Argamon et al. are simply estimating the example disagreement. They also point out that committee-based selection can be viewed as a Monte Carlo method for estimating label distributions over all possible models, given the labeled data.

Abe and Mamitsuka (1998) use a bagging and boosting approach for maximizing the classifier accuracy on the test data. This approach suggests that by maximizing the margin on training data, accuracy on test data is improved, an approach that is not always successful (Grove & Schuurmans, 1998). Furthermore, like the QBC algorithms before it, the QBC-by-boosting approach fails to maximize the margin on *all* unlabeled data, instead choosing to query the single instance with the smallest margin.

McCallum and Nigam (1998b) extend the earlier QBC approach by not only using pool-based QBC, but also using a novel disagreement metric. Whereas the stream-based approaches classify whenever a level of disagreement (possibly any) occurs, in pool-based QBC, the *best* unlabeled example is chosen. Argamon-Engelson and Dagan (1999) suggest using a probabilistic measure based on vote-entropy of the committee, whereas McCallum & Nigam explicitly measure disagreement using the Jensen-Shannon divergence (Lin, 1991; Pereira et al., 1993). However, they recognize that this error metric does not measure the impact that a labeled document had on classifier uncertainty on *other* unlabeled documents. They therefore factored document density into their error metric, to decrease the likelihood of uncertain documents that are outliers. Nevertheless, document density is a rough heuristic that is specific to text classification, and does not directly measure the impact of a document's *label* on other predicted labelings.

More recently, Tong and Koller (2000a) use active learning with Support Vector Machines for text classification. Their SVM approach reduces classifier uncertainty by estimating the reduction in version space size as a function of querying instances. Thus, like QBC, they explicitly reduce version space size, implicitly reducing future expected error. The active learning technique they propose also makes strong assumptions about the linear separability of the data.

Similar in approach to our work, Lindenbaum et al. (1999) examine active learning by minimizing the expected error using nearest neighbor classifiers. Their approach is very similar to ours with respect to loss function; the maximization of expected utility is exactly equivalent to our minimization of error with a 0/1 loss function. However, they do not smooth label distributions by using bagging.

Tong and Koller (2000b) describe a method of active learning for learning the parameters of Bayes nets. Their "expected posterior risk" is very similar to our expected error as in equation (1). However, they use a slightly different loss function and average the loss over the possible models, as opposed to estimating the loss of the maximum *a posteriori* distribution itself. Their method emphasizes learning a good joint distribution over the instance space, which has the advantage of creating better generative models, but may not necessarily lead to the most useful queries for a discriminative model.

5. Experimental Results

NEWSGROUP DOMAIN

The first set of experiments used Ken Lang's *Newsgroups*, containing 20,000 articles evenly divided among 20 UseNet discussion groups (McCallum & Nigam, 1998b). We performed two experiments to perform binary classification. The first experiment used the two classes `comp.graphics` and `comp.windows.x`. The data was pre-processed to remove UseNet headers and UU-encoded binary data. Words were formed from contiguous sequences of alphabetic characters. Additionally, words were removed if they are in a stoplist of common words, or if they appear in fewer than 3 documents. As in McCallum and Nigam (1998b), no feature selection or stemming was performed. The resulting vocabulary had 10,205 words. All results reported are the average of 10 trials. The data set of 2000 documents was split into a training set of 1000 documents, and 1000 test documents.

We tested 4 different active learning algorithms:

- *Random* – choosing the query document at random.
- *Uncertainty Sampling* – choosing the document with the largest label uncertainty, as in (Lewis & Gale, 1994).
- *Density-Weighted QBC* – choosing the document with the greatest committee disagreement in the predicted label, as measured by Jensen-Shannon divergence, weighted by document density, as in (McCallum & Nigam, 1998b). The number of committees used is three.
- *Error-Reduction Sampling* – the method introduced in this paper – choosing the document that maximizes the reduction in the total predicted label entropy, as in equation (1), with error as given in equation (3).[2] The number of bags used is three.

The algorithms were initially given 6 labeled examples, 3 from each class.

At each iteration, 250 documents (25% of the unlabeled documents) were randomly sampled from the larger pool of unlabeled documents as candidates for labeling.[3] The error metric was then computed for each putative labeling against all remaining unlabeled documents (not just the sampled pool.) Figure 1 shows the active learning process. The vertical axis show classification accuracy on the held-out test set, up to 100 queries. All results reported are the average of 10 trials.

The solid gray line at 89.2% shows the maximum possible accuracy after all the unlabeled data has been labeled. After 16 queries, the *Error-Reduction Sampling algorithm* reached 77.2%, or 85% of the maximum possible accuracy. The *Density-Weighted QBC* took 68 queries to reach the same point (four times more slowly), and maintained a lower accuracy for the remainder of the queries.

It is also interesting to compare the documents chosen by the two algorithms for initial labeling. Looking at the documents chosen in the first 10 queries, over the 10 trials, the first 10 documents chosen by the *Error-Reduction Sampling* algorithm were an FAQ, tutorial or HOW-TO 9.8 times out of ten. By comparison, the first 10 documents chosen by the *Density-Weighted QBC* algorithm were an FAQ or HOW-TO only 5.8 times out of 10. While the high incidence of the highly informative documents in the initial phases is not quantitatively meaningful, it does suggest that the learner's behavior is somewhat intuitive.

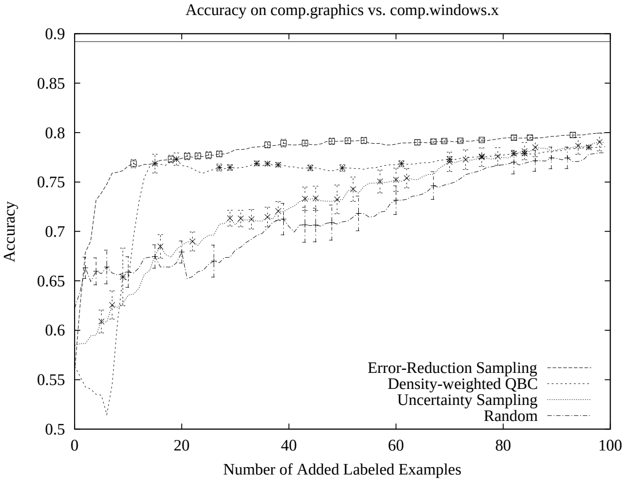

Figure 1. Average test set accuracy for `comp.graphics` vs. `comp.windows.x`. The *Error-Reduction Sampling* algorithm reaches 85% of maximum in 16 documents, compared to 68 documents for the *Most Disagreed* algorithm. The error bars are placed at local maximum to reduce clutter.

The particular newsgroups in the preceding experiment were chosen because they are relatively easy to distinguish. A more difficult text-categorization problem is classifying the newsgroups `comp.sys.ibm.pc.hardware` and `comp.os.ms-windows.misc`. The documents were pre-processed as before, resulting in a vocabulary size of 9,895. The data set of 2000 documents was split into a training set of 1000 documents, and 1000 test documents. Also as before, the unlabeled data was sampled randomly down to 250 documents for candidate labelings at each iteration, although the sampling error was measured against all unlabeled documents.

We can again examine the documents chosen by the different algorithms during the initial phases. *Error-Reduction Sampling* had an average incidence of 7.3 FAQs in the first 10 documents, compared with 2.6 for *Density-Weighted QBC*. In this experiment, however, we see that the intuitive behavior is not sufficient for one algorithm to clearly out-perform another, and the learners required several more documents to begin to achieve a reasonable accuracy.

The solid gray line at 88% shows the maximum possible accuracy after all the unlabeled data has been labeled. After 42 queries, the *Error-Reduction Sampling algorithm* reached 75%, or 85% of the maximum possible accuracy. The *Density-Weighted QBC* algorithm reached the same accuracy after 70 queries, or 1.6 times more slowly.

JOB CATEGORY DOMAIN

The third set of experiments used a data set collected at WhizBang! Labs. The *Job Category* data set contained 112, 643 documents containing job descriptions for 16 different categories such as `Clerical`, `Educational` or `Engineer`. The 16 different categories were then bro-

[2] We also tried *Error-Reduction Sampling* with 0/1 loss, but performance was essentially random. As of yet, we have no explanation.

[3] The sub-sampling was performed in the interests of these experimental results. In a real active learning setting, all algorithms would be run over as much unlabeled data as was computationally feasible in that setting.

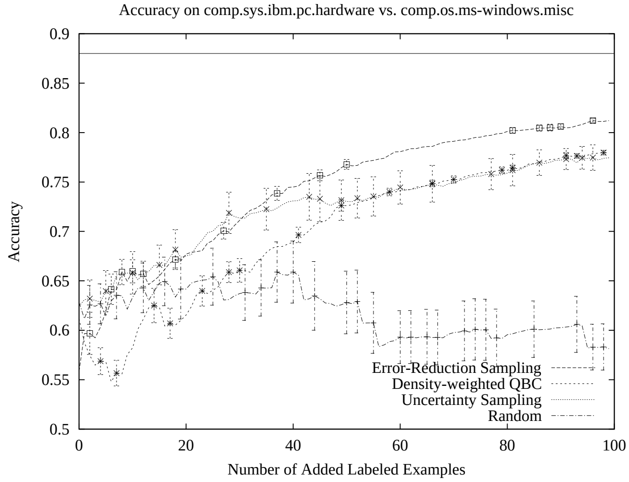

Figure 2. Average test set accuracy for the `comp.sys.ibm.-pc.hardware` vs. `comp.os.ms-windows.misc`. The *Error-Reduction Sampling* algorithm reaches 85% of maximum in 45 documents, compared to 72 documents for the *Density-Weighted QBC* algorithm. The error bars again are placed at local maximum.

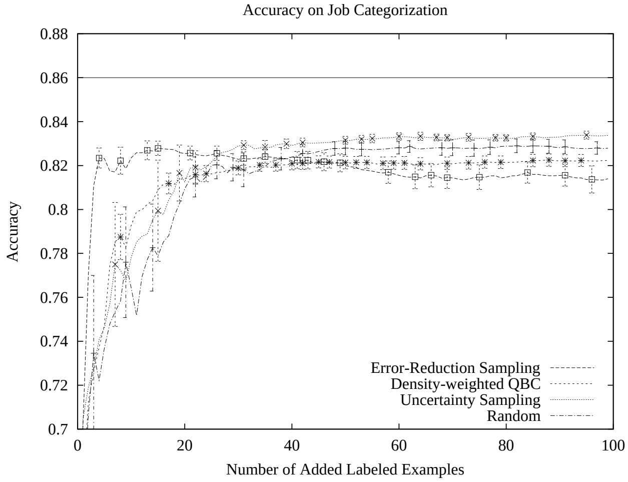

Figure 3. Average test set accuracy for the *Job Category* domain, distinguishing one job category from 5 others. The *Error-Reduction Sampling* algorithm reaches 82% accuracy in 5 documents, compared to 36 documents for both the *Density-Weighted QBC* and *Random* algorithms.

ken down into as many as 9 subcategories. The data set was collected by automatic spidering company job openings on the web, and labeled by hand. We selected the `Engineer` job category, and took 500 articles from each of the six `Engineer` categories: `Chemical`, `Civil`, `Industrial`, `Electrical`, `Mechanical`, `Operations` and `Other`. The documents were pre-processed to remove the job title, as well as rare and stoplist words.

This experiment trained the naive Bayes classifier to distinguish one job category from the remaining five. Each data point is an average of 10 trials per job category, averaged over all 6 job categories. In this example, the *Error-Reduction Sampling* algorithm reached 82% accuracy (94% of the maximum accuracy at 86%) in 5 documents. The *Job Category* data set is easily distinguishable, however, since similar accuracy is achieved after choosing 36 documents at random. The region of interest for evaluating this domain is the initial stages as shown by figure 4. Although the other algorithms did catch up, the *Error-Reduction Sampling* algorithm reached very high accuracy very quickly.

6. Summary

Unlike earlier work in version-space reduction, our approach aims to maximize expected error reduction directly. We use the pool of unlabeled data to estimate the expected error of the current learner, and we determine the impact of each potential labeling request on the expected error. We reduce the variance of the error estimate by averaging over several learners created by sampling (bagging) the labeled data. This approach can be compared to existing statistical techniques (Cohn et al., 1996) that compute the reduction in error (or some equivalent quantity) in closed form; however, we approximate the reduction in error by repeated sampling. In this respect, we have attempted to bridge the gap between closed-form statistical active learning and more recent work in Query-By-Committee (Freund et al., 1997; McCallum & Nigam, 1998b).

We presented results on two domains, the *Newsgroups* domain and also the *Job Category* domain. Our results show that *Error-Reduction Sampling* algorithm outperforms some existing algorithm substantially, achieving a high level of accuracy with fewer than 25% of the labeling queries required by *Density-Weighted QBC*.

A naive implementation of our approach is computationally complex compared to most existing QBC algorithms. However, we have shown a number of optimizations and approximations that make this algorithm much more efficient and tractable. Ultimately, the trade-off between computational complexity and the number of queries should always be decided in favor of fewer queries, each of which requires humans in the loop. A human labeler typically requires 30 seconds or more to label a document, during which time a computer active learner can select an example in a very large pool of documents. The results presented here typically required less than a second of computation time per query.

Furthermore, our algorithm uses sub-sampling of the unlabeled data to generate a pool of candidates at each iteration. By initially using a fairly restrictive pool of candidates for labeling, and increasing the pool as time permits, our algorithm can be considered an anytime algorithm.

Our technique should perform even more strongly with

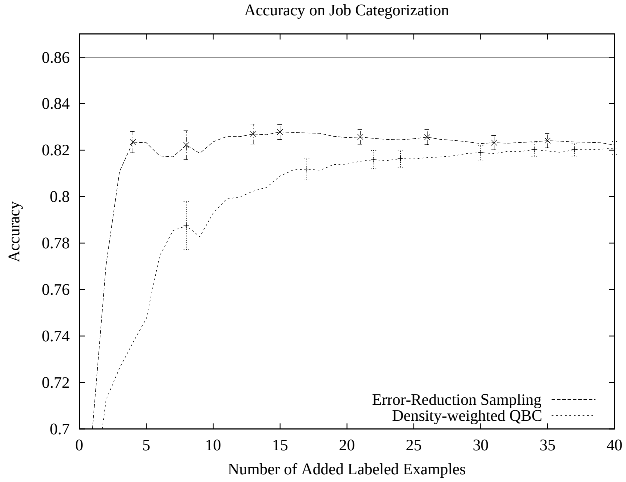

Figure 4. A magnified view of the average test set accuracy for the *Job Category* domain, distinguishing one job category from 5 others. The *Error-Reduction Sampling* algorithm clearly reaches a high level of accuracy much before the *Density-Weighted QBC* algorithm.

models that are complex and have complex parameter spaces, not all regions of which are relevant to performance on a particular data set. In these situations active learning methods based on version space reduction would not be expected to work as well, since they will expend effort excluding portions of version space that have no impact on expected error.

We plan to extend this active learning technique to other classifiers, such as Support Vector Machines. The recent work by Cauwenberghs and Poggio (2000) describes techniques for efficient incremental updates to SVMs and will apply to our approach.

Acknowledgements

Both authors were supported by Whizbang! Labs for this work. The authors would like to thank Andrew Ng for his advice and suggestions. Fernando Pereira, Kamal Nigam, Simon Tong and Pedro Domingos provided much valuable feedback and suggestions of earlier versions of this paper, as did the anonymous reviewers.

References

Abe, N., & Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. *International Conference on Machine Learning (ICML).*

Argamon-Engelson, S., & Dagan, I. (1999). Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, *11*, 335–460.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Cauwenberghs, G., & Poggio, T. (2000). Incremental and decremental support vector machine learning. *Advances in Neural Information Processing 13 (NIPS).* Denver, CO.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Domingos, P. (2000). Bayesian averaging of classifiers and the overfitting problem. *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 223–230).

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

Grove, A., & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98).*

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the European Conference on Machine Learning.*

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

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. *Proceedings of the National Conference in Artificial Intelligence (AAAI-97).* Providence, RI.

Lin, K. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory, 97*, 145–151.

Lindenbaum, M., Markovitch, S., & Rusakov, D. (1999). Selective sampling for nearest neighbor classifiers. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)* (pp. 366–371).

McCallum, A., & Nigam, K. (1998a). A comparison of event models for naive Bayes text classification. *AAAI-98 Workshop on "Learning for Text Categorization".*

McCallum, A., & Nigam, K. (1998b). Employing EM and pool-based active learning for text classification. *Proc. of the Fifteenth Inter. Conference on Machine Learning* (pp. 359–367).

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*.

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. *Proceedings of the IJCAI-99 workshop on information filtering.*.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of English words. *Proceedings of the 31st ACL.*

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

Tong, S., & Koller, D. (2000a). Support vector machine active learning with applications to text classification. *Proc. of the Seventeenth International Conference on Machine Learning.*

Tong, S., & Koller, D. (2000b). Active learning for parameter estimation in Bayesian networks. *Advances in Neural Information Processing 13 (NIPS).*