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 be an unknown conditional distribution over inputs, , and output classes, , and let be the marginal "input" distribution. The learner is given a labeled training set, , consisting of IID input/output pairs drawn from , and estimates a classification function that, given an input , produces an estimated output distribution . We can then write the expected error of the learner as follows:

(1)

where is some loss function that measures the degree of our disappointment in any differences between the true distribution, and the learner's prediction, .
Two common loss functions are:
log loss:
and 0/1 loss:

.

First-order Markov active learning thus aims to select a query, , such that when the query is given label and added to the training set, the learner trained on the resulting set has lower error than any other ,

(2)

We concern ourselves here with *pool-based active learning*, in which the learner has available a large pool, , of unlabeled examples sampled from , 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 .

This paper takes a sampling approach to error estimation and the choice of query. Rather than estimating expected error over the full distribution, , we measure it over the sample in the pool. Furthermore, the true output distribution is unknown for each sample , 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 as , for log loss we have

— (3)

and for 0/1 loss

— (4)

Of course, before we make the query, the true label for is also unknown. Again, the current learned classifier gives us an estimate of the distribution from which the 's true label would be chosen, , and we can use this in an expectation calculation by calculating the estimated error for each possible label, , and taking an average weighted by the current classifier's posterior, of .

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, , in the pool as a candidate for the next labeling request
      i. consider each possible label, , for , and add the pair to the training set
      ii. re-train the classifier with the enlarged training set,
      iii. estimate the resulting expected loss as in equation (3) or equation (4).
   (b) Assign to the average expected losses for each possible labeling, , weighted according to the current classifier's posterior,
2. Select for labeling the unlabeled example 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.