

2006

Active Learning to Maximize Area Under the ROC Curve

Matt Culver

University of Nebraska at Lincoln, mculver@cse.unl.edu

Deng Kun

University of Nebraska at Lincoln, kdeng@cse.unl.edu

Stephen Scott

University of Nebraska - Lincoln, sscott2@unl.edu

Follow this and additional works at: <http://digitalcommons.unl.edu/cseconfwork>



Part of the [Computer Sciences Commons](#)

Culver, Matt; Kun, Deng; and Scott, Stephen, "Active Learning to Maximize Area Under the ROC Curve" (2006). *CSE Conference and Workshop Papers*. 123.

<http://digitalcommons.unl.edu/cseconfwork/123>

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

Active Learning to Maximize Area Under the ROC Curve

Matt Culver, Deng Kun, and Stephen Scott
Dept. of Computer Science
256 Avery Hall
University of Nebraska
Lincoln, NE 68588-0115
{mculver,kdeng,sscott}@cse.unl.edu

Abstract

In active learning, a machine learning algorithm is given an unlabeled set of examples U , and is allowed to request labels for a relatively small subset of U to use for training. The goal is then to judiciously choose which examples in U to have labeled in order to optimize some performance criterion, e.g. classification accuracy. We study how active learning affects AUC. We examine two existing algorithms from the literature and present our own active learning algorithms designed to maximize the AUC of the hypothesis. One of our algorithms was consistently the top performer, and Closest Sampling from the literature often came in second behind it. When good posterior probability estimates were available, our heuristics were by far the best.

1 Introduction

In active learning, a learning algorithm A is given an unlabeled set of examples U , and is allowed to request labels for a relatively small subset of U to use for training. The goal of active learning is to choose which examples in U to have labeled in order to optimize some performance criterion, e.g. classification accuracy. Applications of active learning include those in which unlabeled data are plentiful and there are only enough resources to label relatively few of them. Such data sets include web pages, biological sequences, and images.

ROC (Receiver Operating Curve) analysis has attracted high attention in machine learning research in the last few years. Due to its robustness in imprecise environments, ROC curves have been advocated and gradually adopted as an alternative to classical machine learning metrics such as misclassification rate. ROC has long been used in other fields such as in signal detection [10] and medical diagnosis [11] to describe the trade-off between true positive rate

(TPR) and true negative rate (TNR) of a two-class classification model. In machine learning, people usually use TPR vs. false positive rate (FPR) to generate the plot. The area under the ROC curve (AUC) is of particular interest in that both strong statistical properties and real world experiments have indicated it superior to empirical misclassification rate as an evaluation measure [7]. AUC has also been used as a tool to select or construct models [15, 22, 21]. When the class distribution and cost functions are skewed or unknown, significant advantages have been observed.

TPR and FPR depend on the classifier function h and the threshold θ used to convert $h(x)$ to a binary prediction. One thus plots TPR vs. FPR as a curve by varying the threshold θ , resulting in the ROC curve. The area under the curve (AUC) indicates the performance of this classifier: the larger the better (an AUC of 1.0 indicates a perfect ranking). There is also a strong connection between AUC and a statistical ranking test: AUC is the same as the Wilcoxon-Mann-Whitney statistic [11], which is an unbiased probability estimate that a randomly drawn positive example would be ranked higher than a randomly drawn negative example.

We study how active learning affects AUC. We examine two existing algorithms from the literature: Roy and McCallum's [23] Error Reduction Sampling (ERS) algorithm (designed to directly minimize prediction error) and the "Closest Sampling" method [24, 26] (sampling closest to the decision boundary; well-known as a simple, fast, high-performance active sampler). We also present our own active learning algorithms designed to maximize the AUC of the hypothesis. One of our algorithms (ESTAUC) was consistently the top performer, and Closest often came in second behind it. When good posterior probability estimates were available, ESTAUC and another of our heuristics (RAR) were by far the best.

The rest of this paper is organized as follows. In Section 2 we review related work. Then in Section 3 we present our algorithms and compare them to ERS, Closest, and random sampling in Section 4. We conclude in Section 5.

2 Related Work

In order to achieve labeling efficiency, an active learner tries to select the most informative example from the unlabeled pool U with respect to some performance measure. A typical performance measure that has been extensively studied in active learning is expected accuracy. One basic idea [6] is to select examples that effectively shrink the current version space (the set of hypotheses consistent with the training data). As discussed by Tong and Koller [26], such a heuristic would be probabilistically optimal if it could always exactly halve the version space each time A makes a query. Strong theoretical results using this idea are yielded by the Query by Committee algorithm [9, 25].

The ‘‘Closest Sampling’’ method [24, 26] (sometimes called ‘‘uncertainty sampling’’) can be thought of as a heuristic to shrink the version space. It greedily selects points that are closest to the current decision boundary. The intuition behind this is that points that are closest to the current decision boundary are points that algorithm A is most uncertain about. By labeling these, A can have better knowledge about the correctness of its decision. Tong and Koller [26] explain why this method often works: in a large margin classifier such as a support vector machine (SVM), the current hypothesis lies approximately ‘‘in the center’’ of the version space and by choosing an example (a hyperplane in version space) that is closest to it, it also cuts the version space approximately in half. However, if all the candidate points are very close to or lying on the decision hyperplane, it seems reasonable to explore diversity among these examples [4, 20]. Also, focusing on examples near the decision boundary prevents exploration of the feature space for regions of examples that the current hypothesis misclassifies [2, 19].

Another approach [9, 12, 14, 23] is to select examples that are helpful in building up confidence in low future error. It is impossible to know the exact future error without knowing the target concept, but approximations make this method feasible. For example, Roy and McCallum [23] suggest to directly minimize the expected error on the dataset. They started by fixing a loss function and then estimated the change in loss of the classifier when a candidate example $x \in U$ and its label were added to L . Specifically, when log loss is used, Roy and McCallum’s algorithm would choose

$$\operatorname{argmin}_{x \in U} \left\{ - \sum_{x' \in U} \sum_{y' \in Y} P(y' | x') \log \hat{P}_{xy_x}(y' | x') \right\},$$

where y_x is the true label of example x , Y is the set of labels, $P(y' | x')$ is the true posterior probability of label y' given instance x' , and $\hat{P}_{xy_x}(y' | x')$ is the estimate of the posterior by the model trained on $L \cup \{(x, y_x)\}$. Since

$P(y' | x')$ is unknown, Roy and McCallum used their estimate \hat{P} in its place. Since y_x is also unknown, they considered adding x with each label individually, then combined the two loss estimates weighted by their posterior estimate. Thus for log loss, they selected

$$\operatorname{argmin}_{x \in U} \left\{ - \sum_{y \in Y} \hat{P}(y | x) \sum_{x' \in U} \sum_{y' \in Y} \hat{P}_{xy}(y' | x') \log \hat{P}_{xy}(y' | x') \right\},$$

where $\hat{P}(y | x)$ is the posterior estimate of the current model (i.e. the one trained on L). Because the candidate model’s posterior estimate is used in place of the true posterior probability in the loss function, ERS selects those examples that maximize the sharpness of the learner’s posterior belief about the unlabeled examples [23].

Related to Roy and McCallum’s work, Nguyen and Smeulders [18] chose examples that have the largest contribution to the current expected error: they built their classifiers based on centers of clusters and then propagated the classification decision to the other samples via a local noise model. During active learning, the clustering is adjusted using a coarse-to-fine strategy in order to balance between the advantage of large clusters and the accuracy of the data representation. Yet another approach [17] is specific to SVM learning. Conceptually, in SVM learning if we can find all the true support vectors and label all of them, we will guarantee low future error. Mitra et al. assigned a confidence factor c to examples within the current decision boundary and $1 - c$ to examples outside each indicating the confidence of whether they are true support vectors, and then chose those examples probabilistically according to this confidence.

The third category of active learning approaches contains active learning algorithms that try to quickly ‘‘boost’’ or ‘‘stabilize’’ an active learner. Active learning is unstable, especially with limited labeled examples, and the hypothesis may change dramatically each round it sees a new example. One way to boost active learning algorithms is simply combining them in some way. For example, the algorithm COMB [2] combines three different active learners by finding and fast-switching to the one that currently performs the best. Osugi et al. [19] adopted a similar approach with a simpler implementation and focused on how to balance the exploration and exploitation of an active learner. In their implementation the empirical difference between the current hypothesis and the previous one is used as a criterion to decide whether exploration should be further encouraged. In other exploration-based active learning, Xiao et al. [28] studied the problem of active learning in extracting useful information in commercial games, in which ‘‘decision-boundary refinement sampling’’ (analogous to Closest sampling) and ‘‘default rule sampling’’ (analogous to random sampling) mechanisms are each used half of the time.

Since each of the above algorithms attempts to directly optimize some measure of performance (e.g. minimizing uncertainty or minimizing future prediction error), one would expect such algorithms to tend to increase AUC as a side effect. The purpose of our work is to assess how well some algorithms do just that, as well as presenting algorithms designed to directly maximize AUC.

3 Our Algorithms

We now describe our active learning algorithms designed to maximize AUC. Throughout this section, we let $h(x) \in \mathbb{R}$ denote the current hypothesis's confidence that example x is positive (the larger the value, the higher the confidence in a positive label). The value of h need not be a probability estimate except in one of our algorithms (ESTAUC).

In our first heuristic, **Rank Climbing** (RANC), we use the current hypothesis h to rank all examples in $L \cup U$ where L is the set of labeled examples used to train h , and U is the unlabeled pool. The examples are ranked in descending order according to the confidences $h(x)$. Let x_ℓ be the lowest ranked example from L with a positive label. We select for labeling the lowest ranked unlabeled example that is ranked higher than x_ℓ :

$$\operatorname{argmin}_{x \in U: h(x) > h(x_\ell)} \{h(x)\} .$$

In the unlikely event that there is no example in U ranked higher than x_ℓ , RANC chooses the highest-ranked example from U .

In our second heuristic, **Rank Sampling** (RANS), we again use the current hypothesis h to rank all examples in $L \cup U$ in descending order according to their confidences. Let x_u be the highest ranked example from L with a negative label, and x_ℓ be the lowest ranked example from L with a positive label. The example that we choose to label is selected uniformly at random from the set $C = \{x \in U : h(x) > h(x_\ell) \text{ and } h(x) < h(x_u)\}$. If C is empty then we repeatedly change x_u to be the next highest ranked example from L , and x_ℓ to be the next lowest ranked example from L until C is non-empty.

Since AUC is proportional to the number of negative examples ranked below positive examples, RANC and RANS attempt to find an unlabeled negative example that ranks above positive examples. RANC assumes that the lowest-ranked example above x_ℓ is the most likely to be negative, while RANS makes a random selection to reduce sensitivity to noise.

For our next algorithm, first assume that we know the labels of the examples in U . Then the example from U that we choose to label is the one that most improves the AUC on the unlabeled examples, where the AUC is computed by

the formula of Hanley and McNeil [11] given below. More precisely, we would choose

$$\operatorname{argmax}_{x \in U} \left\{ \frac{\sum_{x' \in U} \sum_{x'' \in U(h_{xy_x}, x')} I(y_{x'} = +) I(y_{x''} = -)}{|P| |N|} \right\}$$

where $y_{x'}$ is the true label of example x' , $h_{xy_x}(x')$ is the confidence of the hypothesis trained on $L \cup \{(x, y_x)\}$ and evaluated on x' , $U(h_{xy_x}, x') = \{x'' \in U : h_{xy_x}(x'') < h_{xy_x}(x')\}$ is the subset of examples in U that have confidence less than that of x' when evaluated with h_{xy_x} , $P = \{x \in U : y_x = +\}$, $N = \{x \in U : y_x = -\}$, and $I(\cdot) = 1$ if its argument is true and 0 otherwise. Since the denominator is independent of x , we instead can use the unnormalized AUC:

$$\operatorname{argmax}_{x \in U} \left\{ \sum_{x' \in U} \sum_{x'' \in U(h_{xy_x}, x')} I(y_{x'} = +) I(y_{x''} = -) \right\} . \quad (1)$$

Since we do not know the true labels of the examples in U , we adapt the approach of Roy and McCallum [23] and use probability estimates derived from the hypothesis h_x in place of the indicator functions. Further, since we do not yet know the label of the candidate point we are considering labeling, we compute (1) using each possible label and weight them according to our posterior probability estimates of each label:

$$\operatorname{argmax}_{x \in U} \left\{ \hat{P}(+ | x) \sum_{x' \in U} \sum_{x'' \in U(h_{x+}, x')} \hat{P}_{x+}(+ | x') \hat{P}_{x+}(- | x'') \right. \\ \left. + \hat{P}(- | x) \sum_{x' \in U} \sum_{x'' \in U(h_{x-}, x')} \hat{P}_{x-}(+ | x') \hat{P}_{x-}(- | x'') \right\}$$

where $h_{x+}(x')$ is the confidence of the hypothesis trained on $L \cup \{(x, +)\}$ and evaluated on x' . In addition, $\hat{P}(y | x)$ is the probability of predicting $y \in \{+, -\}$ given x by hypothesis h trained on L , and $\hat{P}_{xy}(y' | x')$ is the probability of predicting y' given x' by hypothesis h_{xy} trained on $L \cup \{(x, y)\}$. The probability estimates may come from e.g. naïve Bayes or from logistic regression with an SVM. We refer to this approach as **Maximizing Estimated AUC** (ESTAUC).

Our fourth heuristic, **Rank Reinforcing** (RAR), focuses on the ranking induced by the hypothesis. RAR uses the current model h to label all examples in U and uses this labeling in the computation of the AUC of U as ranked by

h_{x+} and h_{x-} . Specifically, the example RAR chooses is

$$\operatorname{argmax}_{x \in U} \left\{ \hat{P}(+ | x) \sum_{x' \in U} \sum_{x'' \in U(h_{x+}, x')} f(x', x'') \right. \\ \left. + \hat{P}(- | x) \sum_{x' \in U} \sum_{x'' \in U(h_{x-}, x')} f(x', x'') \right\},$$

where $f(x', x'') = I(h(x') > \theta) I(h(x'') < \theta)$ and θ is the threshold used to map $h(\cdot)$ to a binary label. Ties can be broken as follows. Let $T \subseteq U$ be the set of examples involved in the tie. We can break ties either by applying ESTAUC over T or by summing the margins of pairs of relevant examples in T :

$$\operatorname{argmax}_{x \in T} \left\{ \hat{P}(+ | x) \sum_{x' \in U} \sum_{x'' \in U(h_{x+}, x')} f(x', x'')(h(x') - h(x'')) \right. \\ \left. + \hat{P}(- | x) \sum_{x' \in U} \sum_{x'' \in U(h_{x-}, x')} f(x', x'')(h(x') - h(x'')) \right\}.$$

RAR amounts to choosing $x \in U$ such that h_x most reinforces h 's ranking of the unlabeled examples to the extent that examples that h predicts as positive remain ranked higher than examples that h predicts as negative.

If implemented as stated above, the ERS, ESTAUC, and RAR heuristics could be slow due to the need to repeatedly retrain the hypotheses. There are, however, several techniques that allow for speeding up the execution time of these heuristics without harming performance. The first technique is to filter the set of candidate examples that are under consideration for labeling. This can be accomplished through random sampling, or by using a faster active learning heuristic to rank the examples in the unlabeled pool and choosing the most promising ones as the candidates. We found that a candidate pool of 100 examples filtered by Closest Sampling produced very good results for ESTAUC. Using a classifier that is capable of incremental and decremental updates also reduces execution time as it removes the necessity of rebuilding the classifier each time a candidate point is evaluated. For example, both naïve Bayes and SVMs are capable of incremental and decremental updates [5].

4 Experimental Results

4.1 Setup

Experiments were carried out on 8 data sets from the UCI Machine Learning Repository [3], and one dataset derived from the United States Postal Service (USPS) handwritten

digit recognition dataset. (To make the latter data set binary-labeled, from the USPS dataset we only used examples with the digits “3” or “8”.) Information on each dataset is summarized in Table 1, including the ratio of positive to negative examples (P/N). Since AUC was the metric used in evaluating performance, all data sets are two-class.

Table 1. Data set information.

DATA SET	NO. OF INST.	NO. OF ATTR.	P/N
BREAST CANCER	286	9	0.42
COLIC	368	22	0.59
CREDIT A	690	15	0.80
CREDIT G	1000	20	2.33
DIABETES	768	8	1.87
IONOSPHERE	351	34	1.77
KR vs. KP	3196	36	1.09
VOTE	435	16	1.21
USPS	1416	256	1.00

In addition to the four algorithms of Section 3, tests were run with Closest Sampling, Roy and McCallum’s log loss Error-Reduction Sampling (ERS) method, and a random sampler. The heuristics were evaluated using the SVM Sequential Minimal Optimization (SMO) as the base learner. All experiments were run with 15 and 100 examples in the initial labeled training set L_0 . The heuristics were implemented in Java within the Weka machine learning framework [27]. We used the Weka implementation for SMO, applying Weka’s logistic regression to get probability estimates when needed.

We used k -fold cross validation in our tests. Ten folds were used on all of the datasets except Breast Cancer, where seven folds were used due to the small size of the dataset. Our testing methodology is summarized below.

1. For each partition P_i , $i \in \{1, 2, \dots, 10\}$, set aside P_i as the test set, and combine the remaining partitions into the set F_i
 - (a) Repeat 10 times:
 - i. Select an initial labeled training set L_0 of size $m \in \{15, 100\}$ from F_i uniformly at random
 - ii. Use the remainder of F_i as the unlabeled pool U
 - iii. Run each heuristic on L_0 and U , and evaluate on P_i after each query
2. Report the average of the results of all tests.

For ESTAUC, RAR, and ERS, the set of instances under consideration for labeling was reduced to 100 candidates

to increase speed. However, all instances in U were used for estimating AUC. The sized-100 subset of U was chosen as follows. First, the examples in U were ranked by Closest from least to most certain. Then the top 100 most uncertain examples were used as the candidate pool¹. As a control we also introduced a new heuristic called **Random-CS** that selects an example to label uniformly at random from the Closest-filtered candidate set, whereas **Random** chooses uniformly at random from all of U .

Learning curves were constructed to evaluate the behavior of the algorithms. These curves display the change in performance of the heuristics as they make queries. To construct the curves we plotted the AUC achieved by each heuristic on the test set against the size of L after each query is made. The AUC value plotted was the mean over all tests (10 for each fold). AUC was plotted on the y -axis, and the size of the current labeled set was on the x -axis.

Paired- t tests were performed to establish the significance at which the heuristics differ. Using a paired- t test is valid because AUC is approximately normally distributed when the test set has more than ten positive and ten negative examples [13]. We compared all heuristics pairwise at each query, and determined the maximum confidence level at which the difference between them was significant. We used cutoffs at the 0.60, 0.70, 0.80, 0.90, 0.95, 0.975, 0.99, and 0.995 confidence levels. It is not feasible to report the paired- t results for all experiments, but they will be mentioned where appropriate. In addition, they are used in one of our summary statistics.

Significance is established between two heuristics by taking the median confidence level at which they differ across all queries. So for example, if algorithm A has an advantage over B significant at the 0.80 level when $|L| = 20$, an advantage significant at 0.70 when $|L| = 21$, an advantage at 0.90 at $|L| = 22$, no advantage at $|L| = 23$, and if B has an advantage significant at the 0.95 level at $|L| = 24$, then the sorted sequence of significance values for A over B is $(-0.95, 0.0, 0.70, 0.80, 0.90)$. This gives a median advantage of A over B of 0.70. We used the median because it is insensitive to outliers, and because it requires that a heuristic be significantly better on at least half of the queries for it to be considered significantly better overall.

Because we have done a broad analysis of active learning heuristics, there are a large number of results to report. Results are generally displayed using learning curves, but with so many it is difficult to get a handle on the big picture. To aid in this endeavor we also make use of three summary statistics.

The first statistic we refer to as the *ranked performance*

¹We tried varying the size of the candidate pool, with sizes of 50, 100, 150 and 200, and found no increase in performance with more than 100 examples. We also tried simply randomly selecting the candidate pool from U , but performance was adversely affected.

of the heuristics. With this statistic we establish a ranking over heuristics on a dataset taking the paired- t tests into account. We rank each heuristic according to how many of the other heuristics it is significantly better than, based on the median significance level over all queries. With n heuristics the best heuristic will receive a rank of 1 and the worst a rank of n . Therefore, if heuristic A performs significantly worse than heuristic B , but is significantly better than all others, it gets a rank of 2. It is also possible for a heuristic to have a rank *range* rather than a single value. This occurs when the difference between it and another heuristic is not significant. As an example, if heuristic C is significantly worse than two of the heuristics, and there is no significant difference between C and two other algorithms, then C will receive a rank of 3–5. In general, a heuristic A can be considered significantly better than a heuristic B if there is no way for B to be ranked higher than A within the established ranking scheme.

The rank performance statistic is also summarized across all of the datasets by displaying the mean rank and number of wins for each heuristic. An algorithm's mean rank is simply the mean of its lower and upper ranks across all datasets. A win is awarded on each dataset for the heuristic that receives a rank of 1. In the case where multiple heuristics have a 1 in their rank range (i.e. there is no significant difference between them), then partial credit is assigned to each weighted by the width of its rank range. Let $Q = \{q_1, \dots, q_n\}$ be the set of heuristics that have a 1 in their rank range, and $r(q_i)$ be the width of the rank range for heuristic q_i . The win credit earned $W(q_i)$ for heuristic q_i is $W(q_i) = (1/(r(q_i) - 1)) / (\sum_{j=1}^n 1/(r(q_j) - 1))$.

One of the primary aims of active learning is to reduce the amount of training data needed to induce an accurate model. To measure this we define the *target AUC* as the mean AUC achieved by random sampling for the final 20% of the queries. We then report the minimum number of examples needed by each algorithm to achieve the target AUC. We also report the *data utilization ratio*, which is the number of examples needed by each heuristic to reach the target AUC divided by the number needed by random sampling. In the event that a heuristic does not reach the target AUC we simply report that the minimum number of examples needed is greater than the size of L after the last query round. This measure reflects how efficiently a heuristic uses the data, but may not reflect large changes in performance in the later query rounds. This metric is similar to one used by Melville et al. [16] and Abe et al. [1]. To summarize over all datasets we also report the median data utilization ratio and number of wins for each heuristic.

Our last summary statistic is the area under the learning curve above Random, which is the difference between the area under the learning curve for a heuristic and that of Random. A negative value for this statistic indicates that the

heuristic on average performed worse than Random. The area under the learning curve for a heuristic is calculated as the sum of the AUC achieved by a heuristic over all query rounds. It is more sensitive to the overall performance of the heuristics throughout the learning process than the previous two statistics. To summarise across all datasets we also report the mean area above random achieved by a heuristic as well as the number of wins.

4.2 Results

4.2.1 UCI Data

In the first set of experiments, we used SMO as the base classifier and 15 examples in the initial labeled set. ESTAUC performed better than Closest sampling overall. The Random-CS heuristic is also a strong performer, scoring worse than ESTAUC, but better than Closest. RAR and ERS do not perform better than either Closest or Random-CS. Figure 1 shows the learning curves for the Ionosphere dataset. ESTAUC is significantly better than all of the other heuristics from 45 to 94 labeled examples at the 0.6 confidence level or greater. Since it's difficult to discern the finer differences between curves in Figure 1, we look to Tables 2–4 for the summary statistics. ESTAUC is clearly the winner on the ranked performance metric with a win credit that is more than twice that of its nearest competitor. On the data utilization table, Closest receives the same number of wins as ESTAUC, but ESTAUC does better on all of the datasets that Closest doesn't win. On the area under the learning curve, Random-CS actually achieves a slightly higher mean area than ESTAUC even though ESTAUC has the max area on more datasets. Generally, behind ESTAUC, Closest, and Random-CS, we find ERS with RASP close behind, then Random, then RAR, then RANC. Finally, we note that despite the similarities in form between ESTAUC and ERS, the strong differences in performance between them indicate that they are making very different choices of examples to label.

In results not shown, we found little change in the relative performance of the algorithms when starting with 100 examples in the initial labeled set rather than 15. Again ESTAUC was the top performer on all metrics, though it did have more stiff competition from ERS and Closest in this case, mainly due to the fact that there is less room for improvement when starting with 100 labeled examples.

Overall we see that ESTAUC is the best heuristic that we tested on these datasets. While there is an additional cost associated with ESTAUC due to the need to train multiple SVMs, this can be mitigated via the use of incremental and decremental updates and considering only a filtered subset of U . Further, even without an incremental update SVM, the amount of time needed to choose an example to label with ESTAUC (as well as ERS), while greater than that

needed for Closest, was still quite small (10–15 seconds) for most data sets. The only exception was USPS, where ESTAUC and ERS each took about one minute to choose an unlabeled example due to the large number of attributes, while Closest was nearly instantaneous. However, the bulk of this additional time was spent training SMO, so an incremental SVM would mitigate this significantly. Further, even one minute is not at all large relative to the amount of time it takes an oracle to label the example: Consider for instance how long it would take for a human labeler to classify a web page or a biological sequence.

4.2.2 Synthetic Data

Given how ERS, ESTAUC, and RAR are defined with respect to probability estimates, we wanted to see how well they would do if given perfect probability estimates. To do this we developed two synthetic datasets. The synthetic data was created from two overlapping ten-dimensional Gaussian distributions: $\mathcal{N}_+(\mu_+, \sigma^2 I_{10})$ and $\mathcal{N}_-(\mu_-, \sigma^2 I_{10})$, where $\sigma^2 = 12.5$ and I_{10} is the 10×10 identity matrix. For the first dataset (**Gaussian 0,5**), $\mu_+ = (0, \dots, 0)$ and $\mu_- = (5, \dots, 5)$, and for the second dataset (**Gaussian 0,10**), $\mu_+ = (0, \dots, 0)$ and $\mu_- = (10, \dots, 10)$. The classification of each example generated by \mathcal{N}_+ is positive and that of those generated by \mathcal{N}_- is negative. We then ran ESTAUC and RAR with the true Gaussian pdfs P in place of the estimates \hat{P} . We also ran ERS with the true pdf in place of the first $\hat{P}(y' | x')$ term (recall that the second \hat{P} term is the classifier itself, so we did not change it).

Without exception, ESTAUC and RAR dominate the other heuristics on the Gaussian data when using either 15 (shown) or 100 (omitted) initial labeled examples. Figure 2 shows the performance of the active learning heuristics for a representative experiment. Interestingly, ERS did not gain a similar benefit from having perfect probability estimates. The summary statistics for these experiments can be found in Tables 5–7.

These results clearly demonstrate the very strong potential of ESTAUC and RAR if given good probability estimates. However, any probability model that we might generate will necessarily be based on the labeled training data. In active learning this is generally a relatively small set of data. Obviously, it is difficult to generate high quality probability estimates from such a small training set.

5 Conclusions

Area under the ROC curve is an important measure of learning performance. We studied how active learning affects AUC, including studying some algorithms from the literature and introducing four new algorithms for active learning designed to maximize AUC. We evaluated all these

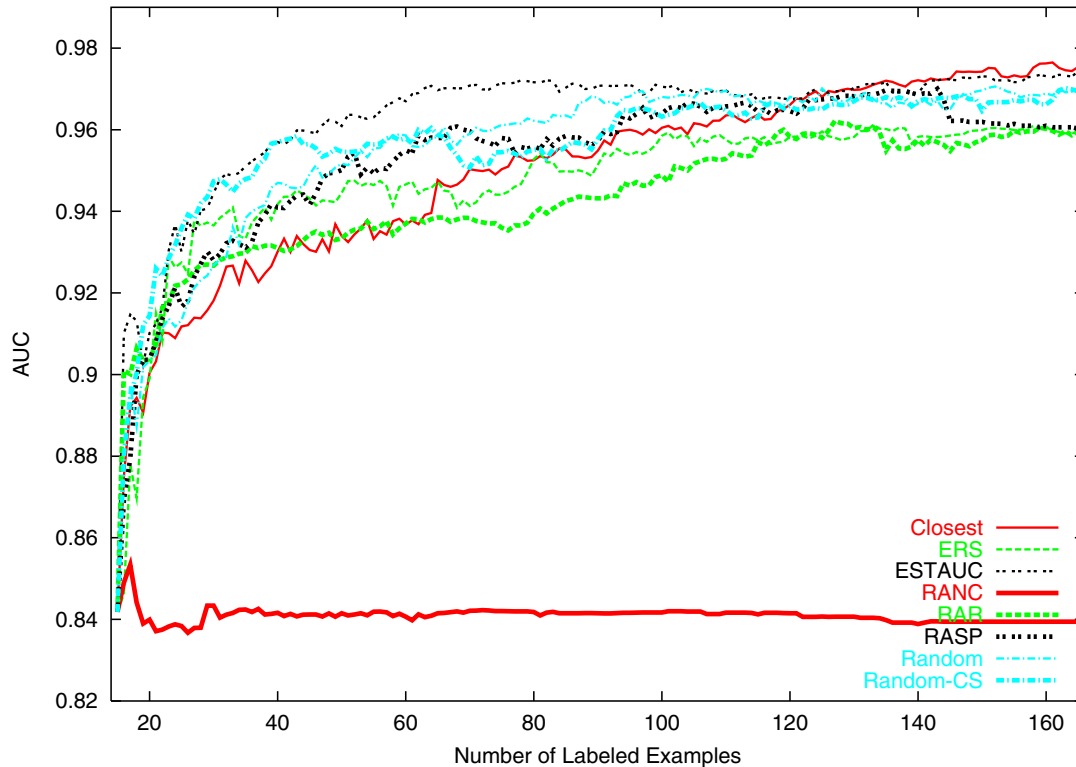


Figure 1. Learning curves for the lonosphere dataset starting with 15 initial labeled points.

algorithms using SMO as the base classifier. Overall, we found that ESTAUC was the top performer. Further, there is strong evidence that if good probability estimates are available, then ESTAUC and RAR will perform very well.

We are currently experimenting with a naïve Bayes base classifier in place of SMO. Future work includes extending this work to multiclass problems and to study the minimization of the lower envelopes of cost curves [8], an alternative to ROC curves.

Acknowledgments

The authors thank Nick Roy for his helpful discussions and the reviewers for their useful comments. This work was funded in part by NSF grant CCR-0092761. This work was completed in part using the Research Computing Facility at the University of Nebraska.

References

[1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the 15th Int. Conf. on Machine Learning*, pages 1–10, 1998.

[2] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291, 2004.

[3] C. L. Blake, D.J. Newman, S. Hettich and C. J. Merz. UCI repository of machine learning databases, 2006. <http://www.ics.uci.edu/~mllearn/MLRepository.html>

[4] K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proc. of the 20th Int. Conf. on Machine Learning*, pages 59–66, 2003.

[5] G. Cauwenberghs and T. Poggio Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, pages 409–415, 2000.

[6] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[7] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems*, Volume 16, 2004.

[8] C. Drummond and R. Holte. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130, 2006.

[9] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[10] D. Green and J. Swets. *Signal detection theory and psychophysics*. John Wiley & Sons, 1964.

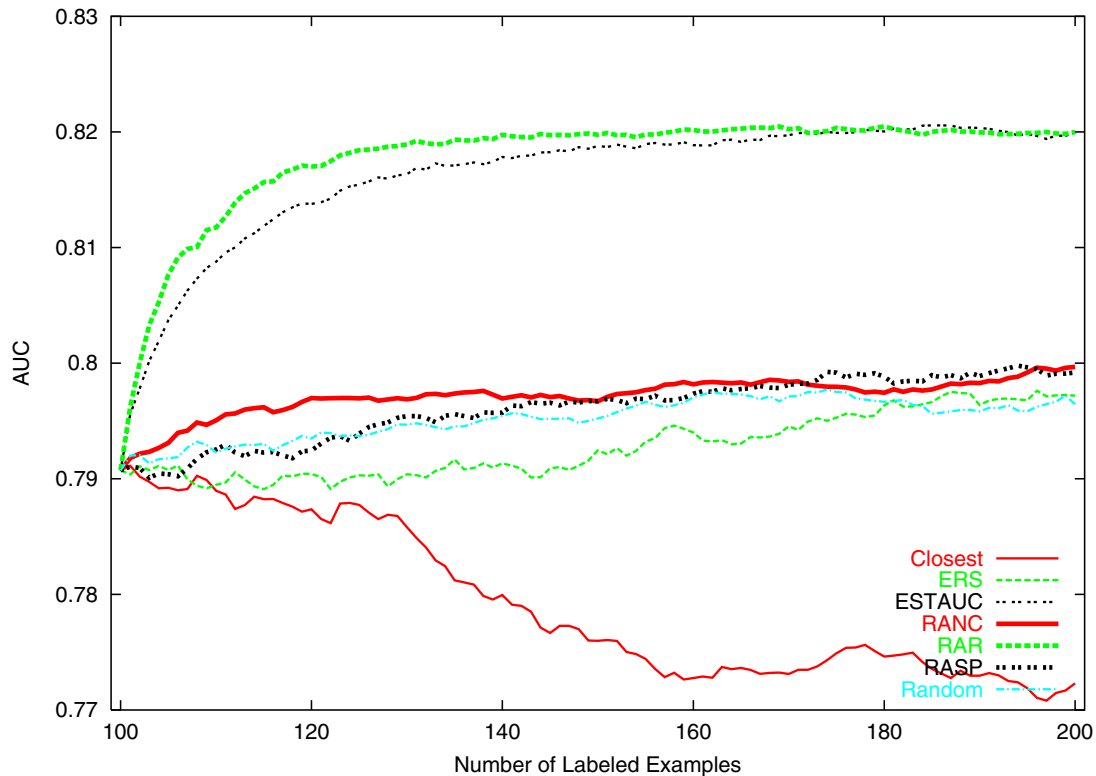


Figure 2. Learning curves for the Gaussian 0,5 dataset.

- [11] J. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.
- [12] V. S. Iyengar, C. Apte, and T. Zhang. Active learning using adaptive resampling. In *Proc. of the 6th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 91–98, 2000.
- [13] E. L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, 1975.
- [14] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54:125–152, 2004.
- [15] D. McClish. Comparing the areas under more than two independent ROC curves. *Med. Decis. Making*, 7:149–155, 1987.
- [16] P. Melville and R. Mooney. Diverse ensembles for active learning. In *Proceedings of the 21st International Conference on Machine Learning*, pages 584–591, 2004.
- [17] P. Mitra, C. A. Murthy, and S. K. Pal. A probabilistic active support vector learning algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(3):413–418, 2004.
- [18] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pages 623–630, 2004.
- [19] T. Osugi, D. Kun, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 330–337, 2005.
- [20] J.-M. Park. Convergence and application of online active sampling using orthogonal pillar vectors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(9):1197–1207, 2004.
- [21] A. Rakotomamonjy. Optimising area under the ROC curve with SVMs. In *ROCAI*, pages 71–80, 2004.
- [22] S. Rosset. Model selection via the AUC. In *Proc. of the 21st Int. Conf. on Machine Learning*, pages 89–96, 2004.
- [23] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning*, pages 441–448, 2001.
- [24] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the 17th Intl. Conf. on Machine Learning*, pages 839–846, 2000.
- [25] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.
- [26] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov):45–66, 2001.
- [27] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.
- [28] G. Xiao, F. Southey, R. C. Holte, and D. Wilkinson. Software testing by active learning for commercial games. In *Proc. of the 20th Nat. Conf. on AI*, pages 898–903, 2005.

Table 2. Ranked performance of the active learning heuristics on UCI data when starting with 15 labeled points.

DATA SET	CLOSEST	ERS	ESTAUC	RANC	RAR	RASP	RANDOM	RANDOM-CS
B. CANCER	2-6	1-3	2-6	1-5	1-5	7-8	6-8	4-7
COLIC	1-6	2-6	1-6	8	7	2-6	2-6	1-3
CREDIT A	5-7	1-4	1-4	8	6-7	5-6	1-4	1-4
CREDIT G	1-4	1-6	1-5	4-8	5-7	2-7	7-8	1-5
DIABETES	1-2	3-7	3-5	4-8	7-8	3-6	5-7	1-2
IONOSPHERE	3-5	6-7	1	8	6-7	2-5	2-4	2-5
KR vs. KP	2-4	2-4	1	8	6	5	7	2-4
USPS	5-6	3-4	1	8	7	2-4	5-6	2-3
VOTE	2	4-8	4-5	6-8	5-8	1	5-7	3
MEAN	2.4-4.7	2.6-5.4	1.7-3.8	6.1-7.7	5.6-6.9	3.3-5.3	4.4-6.3	1.9-4.0
WINS	1.04	0.94	3.71	0.25	0.25	1.00	0.25	1.55

Table 3. Data utilization for the active learning heuristics on UCI data when starting with 15 labeled points. The data utilization ratio (DUR) appears in parentheses below the minimum number of training examples needed to achieve the target AUC.

DATA SET	CLOSEST	ERS	ESTAUC	RANC	RAR	RASP	RANDOM	RANDOM-CS	TARGET AUC
B. CANCER	98 (0.72)	66 (0.48)	75 (0.55)	86 (0.63)	69 (0.50)	161 (1.18)	137 (1.00)	110 (0.80)	0.70
COLIC	113 (0.88)	109 (0.84)	112 (0.87)	>165 (>1.28)	160 (1.24)	123 (0.95)	129 (1.00)	49 (0.38)	0.84
CREDIT A	>165 (>1.11)	>165 (>1.11)	108 (0.72)	>165 (>1.11)	>165 (>1.11)	>165 (>1.11)	149 (1.00)	>165	0.91
CREDIT G	96 (0.64)	106 (0.71)	106 (0.71)	127 (0.85)	115 (0.77)	119 (0.80)	149 (1.00)	104 (0.70)	0.72
DIABETES	91 (0.66)	134 (0.98)	111 (0.81)	133 (0.97)	138 (1.01)	112 (0.82)	137 (1.00)	97 (0.71)	0.81
IONOSPHERE	123 (1.38)	>165 (>1.13)	61 (0.69)	>165 (>1.13)	>165 (>1.13)	128 (1.44)	89 (1.00)	146 (1.64)	0.97
KR vs. KP	83 (0.56)	88 (0.60)	79 (0.54)	>165 (>2.12)	108 (0.73)	108 (0.73)	147 (1.00)	78 (0.53)	0.97
USPS	82 (0.55)	74 (0.50)	44 (0.30)	>165 (>1.11)	>165 (>1.11)	83 (0.56)	149 (1.00)	63 (0.42)	0.99
VOTE	51 (0.37)	104 (0.76)	110 (0.80)	>165 (>1.20)	139 (1.01)	53 (0.39)	137 (1.00)	77 (0.56)	0.99
MEDIAN DUR	0.66	0.76	0.71	>1.11	1.01	0.82	1.00	0.70	
WINS	3	1	3	0	0	0	0	2	

Table 4. Area under the learning curve above Random for the active learning heuristics on UCI data when starting with 15 labeled points.

DATA SET	CLOSEST	ERS	ESTAUC	RANC	RAR	RASP	RANDOM	RANDOM-CS
B. CANCER	1.50	2.37	1.31	2.18	2.07	-0.40	0.00	0.75
COLIC	0.12	-0.41	0.25	-7.68	-2.28	0.26	0.00	1.30
CREDIT A	-0.48	0.10	0.34	-2.05	-1.04	-0.33	0.00	0.27
CREDIT G	2.35	1.56	1.70	0.74	1.09	1.19	0.00	2.04
DIABETES	1.78	0.34	0.75	0.07	-0.90	0.59	0.00	1.33
IONOSPHERE	-0.90	-1.26	1.06	-17.34	-1.92	-0.43	0.00	0.11
KR vs. KP	2.63	2.68	3.29	-18.71	-0.12	1.29	0.00	2.63
USPS	0.02	0.32	0.82	-8.20	-1.18	0.33	0.00	0.45
VOTE	1.11	-0.33	0.03	-0.34	-0.09	1.21	0.00	0.83
MEAN	0.903	0.597	1.061	-5.703	-0.486	0.412	0.00	1.079
WINS	2	1	4	0	0	1	0	1

Table 5. Ranked performance of the active learning heuristics on synthetic data when starting with 15 labeled points.

DATA SET	CLOSEST	ERS	ESTAUC	RANC	RAR	RASP	RANDOM
GAUSSIAN 0,5	7	3	2	6	1	4-5	4-5
GAUSSIAN 0,10	6-7	3-5	1	6-7	2	3-5	3-5
MEAN	6.5-7.0	3.0-4.0	1.5	6.0-6.5	1.5	3.5-5.0	3.5-5.0
WINS	0	0	1	0	1	0	0

Table 6. Data utilization for the active learning heuristics on synthetic data when starting with 15 labeled points. The data utilization ratio (DUR) appears in parentheses below the minimum number of training examples needed to achieve the target AUC.

DATA SET	CLOSEST	ERS	ESTAUC	RANC	RAR	RASP	RANDOM	TARGET AUC
GAUSSIAN 0,5	>165 (>1.57)	66 (0.63)	20 (0.19)	>165 (>1.57)	19 (0.18)	142 (1.35)	105 (1.00)	0.75
GAUSSIAN 0,10	163 (1.21)	116 (0.86)	19 (0.14)	>165 (>1.22)	18 (0.13)	136 (1.01)	135 (1.00)	0.95
MEDIAN DUR	>1.39	0.75	0.17	>1.40	0.16	1.18	1.00	
WINS	0	0	0	0	2	0	0	

Table 7. Area under the learning curve above Random for the active learning heuristics on synthetic data when starting with 15 labeled points.

DATA SET	CLOSEST	ERS	ESTAUC	RANC	RAR	RASP	RANDOM
GAUSSIAN 0,5	-13.87	1.10	7.96	-10.82	9.18	-0.79	0.00
GAUSSIAN 0,10	-5.42	-0.91	3.36	-6.13	2.84	-0.37	0.00
MEAN	-9.645	0.095	5.660	-8.475	6.010	-0.580	0.000
WINS	0	0	1	0	1	0	0