Active Learning for Probability Estimation Using Jensen-Shannon Divergence

Prem Melville[1], Stewart M. Yang[2], Maytal Saar-Tsechansky[4], and Raymond Mooney[3]

[1] Dept. of Computer Sciences, Univ. of Texas at Austin
melville@cs.utexas.edu
[2] windtown@cs.utexas.edu
[3] mooney@cs.utexas.edu
[4] Red McCombs School of Business, Univ. of Texas at Austin
maytal.saar-tsechansky@mccombs.utexas.edu

Abstract. Active selection of good training examples is an important approach to reducing data-collection costs in machine learning; however, most existing methods focus on maximizing classification accuracy. In many applications, such as those with unequal misclassification costs, producing good class probability estimates (CPEs) is more important than optimizing classification accuracy. We introduce novel approaches to active learning based on the algorithms Bootstrap-LV and ACTIVEDECORATE, by using Jensen-Shannon divergence (a similarity measure for probability distributions) to improve sample selection for optimizing CPEs. Comprehensive experimental results demonstrate the benefits of our approaches.

1 Introduction

Many supervised learning applications require more than a simple classification of instances. Often, also having accurate Class Probability Estimates (CPEs) is critical for the task. Class probability estimation is a fundamental concept used in a variety of applications including marketing, fraud detection and credit ranking. For example, in direct marketing the probability that each customer would purchase an item is employed in order to optimize marketing budget expenditure. Similarly, in credit scoring, class probabilities are used to estimate the utility of various courses of actions, such as the profitability of denying or approving a credit application. While prediction accuracy of CPE improves with the availability of more labeled examples, acquiring labeled data is sometimes costly. For example, customers' preferences may be induced from customers' responses to offerings; but solicitations made to acquire customer responses (labels) may be costly, because unwanted solicitations can result in negative customer attitudes. It is therefore critical to reduce the number of label acquisitions necessary to obtain a desired prediction accuracy.

The *active learning* literature [1] offers several algorithms for cost-effective label acquisitions. Active learners acquire training data incrementally, using the model induced from the available labeled examples to identify helpful additional training examples for labeling. Different active learning approaches employ different utility scores to estimate how informative each unlabeled example is, if it is labeled and added to the

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 268–279, 2005.

training data. When successful, active learning methods reduce the number of instances that must be labeled to achieve a particular level of accuracy. Almost all work in active learning has focused on acquisition policies for inducing accurate *classification* models and thus are aimed at improving classification accuracy. Although active learning algorithms for classification can be applied for learning accurate CPEs, they may not be optimal. Active learning algorithms for classification may (and indeed should) avoid acquisitions that can improve CPEs but are not likely to impact classification. Accurate classification only requires that the model accurately assigns the highest CPE to the correct class, even if the CPEs across classes may be inaccurate. Therefore, to perform well, active learning methods for classification ought to acquire labels of examples that are likely to change the rank-order of the most likely class. To improve CPEs, however, it is necessary to identify potential acquisitions that would improve the CPE accuracy, regardless of the implications for classification accuracy. Bootstrap-LV [2] is an active learning approach designed specifically to improve CPEs for binary class problems. The method acquires labels for examples for which the current model exhibits high variance for its CPEs. BOOTSTRAP-LV was shown to significantly reduce the number of label acquisitions required to achieve a given CPE accuracy compared to random acquisitions and existing active learning approaches for classification.

In this paper, we propose two new active learning approaches. In contrast to BOOTSTRAP-LV, the methods we propose can be applied to acquire labels to improve the CPEs of an arbitrary number of classes. The two methods differ by the measures each employs to identify informative examples: the first approach, BOOTSTRAP-JS, employs the Jensen-Shannon divergence measure (JSD) [3]. The second approach, BOOTSTRAP-LV-EXT, uses a measure of variance inspired by the local variance proposed in BOOTSTRAP-LV. We demonstrate that for binary class problems,BOOTSTRAP-JS is at least comparable and often superior to BOOTSTRAP-LV. In addition, we establish that for multi-class problems, BOOTSTRAP-JS and BOOTSTRAP-LV-EXT identify particularly informative examples that significantly improve the CPEs compared to a strategy in which a representative set of examples are acquired uniformly at random. This paper also extends the work of Melville and Mooney [4], which introduced a method, ACTIVEDECORATE, for active learning for classification. They compared two measures for evaluating the utility of examples - label margins and JSD. The results showed that both measures are effective for improving classification accuracy, though JSD is less effective than margins. It was conjectured that JSD would be a particularly useful measure when the objective is improving CPEs. We demonstrate here that, for the task of active learning for CPE, ACTIVEDECORATE using JSD indeed performs significantly better than using margins.

2 Jensen-Shannon Divergence

Jensen-Shannon divergence (JSD) is a measure of the "distance" between two probability distributions [3] which can also be generalized to measure the distance (similarity) between a finite number of distributions [5]. JSD is a natural extension of the Kullback-Leibler divergence (KLD) to a set of distributions. KLD is defined between two distributions, and the JSD of a set of distributions is the average KLD of each

distribution to the mean of the set. Unlike KLD, JSD is a true metric and is bounded. If a classifier can provide a distribution of class membership probabilities for a given example, then we can use JSD to compute a measure of similarity between the distributions produced by a set (ensemble) of such classifiers. If $P_i(x)$ is the class probability distribution given by the i-th classifier for the example x (which we will abbreviate as P_i) we can then compute the JSD of a set of size n as $JS(P_1, P_2, ..., P_n) = H(\sum_{i=1}^{n} w_i P_i) - \sum_{i=1}^{n} w_i H(P_i)$; where w_i is the vote weight of the i-th classifier in the set;[1] and $H(P)$ is the Shannon entropy of the distribution $P = \{p_j : j = 1, ..., K\}$, defined as $H(P) = -\sum_{j=1}^{K} p_j \, log \, p_j$. Higher values for JSD indicate a greater spread in the CPE distributions, and it is zero if and only if the distributions are identical. JSD has been successfully used to measure the utility of examples in active learning for improving classification accuracy [4]. A similar measure was also used for active learning for text classification by McCallum and Nigam [6].

3 Bootstrap-LV and JSD

To the best of our knowledge, Bootstrap-LV [2] is the only active learning algorithm designed for learning CPEs. It was shown to require significantly fewer training examples to achieve a given CPE accuracy compared to random sampling and *uncertainty sampling*, which is an active learning method focused on classification accuracy [7]. Bootstrap-LV reduces CPE error by acquiring examples for which the current model exhibits relatively high local variance (LV), i.e., the variance in CPE for a particular example. A high LV for an unlabeled example indicates that the model's estimation of its class membership probabilities is likely to be erroneous, and the example is therefore more desirable to be selected for learning.

Bootstrap-LV, as defined in [2] is only applicable to binary class problems. We first provide the details of this method, and then describe how we extended it to solve multi-class problems. Bootstrap-LV is an iterative algorithm that can be applied to any base learner. At each iteration, we generate a set of n bootstrap samples [8] from the training set, and apply the given learner $\mathcal{L}$ to each sample to generate n classifiers $C_i : i = 1, ..., n$. For each example in the unlabeled set U, we compute a score which determines its probability of being selected, and which is proportional to the variance of the CPEs. More specifically, the score for example x_j is computed as $(\sum_{i=1}^{n} (p_i(x_j) - \overline{p}_j)^2)/\overline{p}_{j,min}$; where $p_i(x_j)$ denotes the estimated probability the classifier C_i assigns to the event that example x_j belongs to class 0 (the choice of performing the calculation for class 0 is arbitrary, since the variance for both classes is identical), $\overline{p}_j$ is the average estimate for class 0 across classifiers C_i, and $\overline{p}_{j,min}$ is the average probability estimate assigned to the minority class by the different classifiers. Saar-Tsechansky and Provost [2] attempt to compensate for the under-representation of the minority class by introducing the term $\overline{p}_{j,min}$ in the utility score. The scores produced for the set of unlabeled examples are normalized to produce a distribution, and then a subset of unlabeled examples are selected based on this distribution. The labels for these examples are acquired and the process is repeated.

[1] Our experiments use uniform vote weights, normalized to sum to one.

The model's CPE variance allows the identification of examples that can improve CPE accuracy. However as noted above, the local variance estimated by Bootstrap-LV captures the CPE variance of a single class and thus is not applicable to multi class problems. Since we have a set of probability distributions for each example, we can instead, use an information theoretic measure, such as JSD to measure the utility of an example. The advantage to using JSD is that it is a theoretically well-motivated distance measure for probability distributions [3] that can be therefore used to capture the uncertainty of the class distribution estimation; and furthermore, it naturally extends to distributions over multiple classes. We propose a variation of BOOTSTRAP-LV, where the utility score for each example is computed as the JSD of the CPEs produced by the set of classifiers C_i. This approach, BOOTSTRAP-JS, is presented in Algorithm 1.

Our second approach, BOOTSTRAP-LV-EXT, is inspired by the Local Variance concept proposed in BOOTSTRAP-LV. For each example and for each class, the variance in the prediction of the class probability across classifiers $C_i, i = 1, ..., n$ is computed, capturing the uncertainty of the CPE for this class. Subsequently, the utility score for each potential acquisition is calculated as the mean variance across classes, reflecting the average uncertainty in the estimations of all classes. Unlike BOOTSTRAP-LV, BOOTSTRAP-LV-EXT does not incorporate the factor of $\overline{p}_{j,min}$ in the score for multi-class problems, as this is inappropriate in this scenario.

Algorithm 1. Bootstrap-JS

Given: set of training examples T, set of unlabeled training examples U, base learning algorithm $\mathcal{L}$, number of bootstrap samples n, size of each sample m

1. Repeat until stopping criterion is met
2. Generate n bootstrap samples $B_i, i = 1, ..., n$ from T
3. Apply learner $\mathcal{L}$ to each sample B_i to produce classifier C_i
4. For each $x_j \in U$
5. $\forall C_i$ generate CPE distribution $P_i(x_j)$
6. $score_j = JS(P_1, P_2, ..., P_n)$
7. $\forall x_j \in U, D(x_j) = score_j / \sum_j score_j$
8. Sample a subset S of m examples from U based on the distribution D
9. Remove examples in S from U and add to T
10. Return $C = \mathcal{L}(T)$

4 ActiveDecorate and JSD

ACTIVEDECORATE is an active learning method that selects examples to be labeled so as to improve classification accuracy [4]. It is built on the *Query by Committee* (QBC) framework for selective sampling [9]; and has been shown to outperform other QBC approaches, Query by Bagging and Query by Boosting. ACTIVEDECORATE is based on DECORATE [10,11], which is a recently introduced ensemble meta-learner that directly constructs diverse committees of classifiers by employing specially-constructed artificial training examples.

Given a pool of unlabeled examples, ACTIVEDECORATE iteratively selects examples to be labeled for training. In each iteration, it generates a committee of classifiers by applying DECORATE to the currently labeled examples. Then it evaluates the potential utility of each example in the unlabeled set, and selects a subset of examples with the highest expected utility. The labels for these examples are acquired and they are transfered to the training set. The utility of an example is determined by some measure of *disagreement* in the committee about its predicted label. Melville and Mooney [4] compare two measures of utility for ACTIVEDECORATE— *margins* and JSD. Given the CPEs predicted by the committee for an example,[2] the margin is defined as the difference between the highest and second highest predicted probabilities. It was shown that ACTIVEDECORATE using either measure of utility produces substantial error reductions in classification compared to random sampling. However, in general, using margins produces greater improvements. Using JSD tends to select examples that reduce the uncertainty in CPE, which indirectly helps to improve classification accuracy. On the other hand, ACTIVEDECORATE using margins focuses more directly on determining the decision boundary. This may account for its better classification performance. It was conjectured that if the objective is improving CPEs, then JSD may be a better measure.

In this paper, we validate this conjecture. In addition to using JSD, we made two more changes to the original algorithm, each of which independently improved its performance. First, each example in the unlabeled set is assigned a probability of being sampled, which is proportional to the measure of utility for the example. Instead of selecting the examples with the m highest utilities, we sample the unlabeled set based on the assigned probabilities (as in BOOTSTRAP-LV). This sampling has been shown to improve the selection mechanism as it reduces the probability of adding outliers to the training data and avoids selecting many similar or identical examples [12].

The second change we made is in the DECORATE algorithm. DECORATE ensembles are created iteratively; where in each iteration a new classifier is trained. If adding this new classifier to the current ensemble increases the ensemble training error, then this classifier is rejected, else it is added to the current ensemble. In previous work, training error was evaluated using the 0/1 loss function; however, DECORATE can use any loss (error) function. Since we are interested in improving CPE we experimented with two alternate error functions — Mean Squared Error (MSE) and Area Under the Lift Chart (AULC) (defined in the next section). Using MSE performed better on the two metrics used, so we present these results in the rest of the paper. Our approach, ACTIVEDECORATE-JS, is shown in Algorithm 2.

5 Experimental Evaluation

5.1 Methodology

To evaluate the performance of the different active CPE methods, we ran experiments on 24 representative data sets from the UCI repository [13]. 12 of these datasets were

[2] The CPEs for a committee are computed as the simple average of the CPEs produced by its constituent classifiers.

Algorithm 2. ActiveDecorate-JS

Given: set of training examples T, set of unlabeled training examples U, base learning algorithm $\mathcal{L}$, number of bootstrap samples n, size of each sample m

1. Repeat until stopping criterion is met
2. Generate an ensemble of classifiers, $C^* = Decorate(\mathcal{L}, T, n)$
3. For each $x_j \in U$
4. $\forall C_i \in C^*$ generate CPE distribution $P_i(x_j)$
5. $score_j = JS(P_1, P_2, ..., P_n)$
6. $\forall x_j \in U, D(x_j) = score_j / \sum_j score_j$
7. Sample a subset S of m examples from U based on the distribution D
8. Remove examples in S from U and add to T
9. Return $Decorate(\mathcal{L}, T, n)$

two-class problems, the rest being multi-class. For three datasets (*kr-vs-kp*, *sick*, and *optdigits*), we used a random sample of 1000 instances to reduce experimentation time.

All the active learning methods we discuss in this paper are meta-learners, i.e., they can be applied to any base learner. For our experiments, as a base classifier we use a Probability Estimation Tree (PET) [14], which is an unpruned J48[3] decision tree for which Laplace correction is applied at the leaves. Saar-Tsechansky and Provost [2] showed that using Bagged-PETs for prediction produced better probability estimates than single PETs for BOOTSTRAP-LV; so we used Bagged-PETs for both BOOTSTRAP-LV and BOOTSTRAP-JS. The number of bootstrap samples and the size of ensembles in ACTIVEDECORATE was set to 15.

The performance of each algorithm was averaged over 10 runs of 10-fold cross-validation. In each fold of cross-validation, we generated learning curves as follows. The set of available training examples was treated as an unlabeled pool of examples, and at each iteration the active learner selected a sample of points to be labeled and added to the training set. Each method was allowed to select a total of 33 batches of training examples, measuring performance after each batch in order to generate a learning curve. To reduce computation costs, and because of diminishing variance in performance for different selected examples along the learning curve, we incrementally selected larger batches at each acquisition phase. The resulting curves evaluate how well an active learner orders the set of available examples in terms of utility for learning CPEs. As a baseline, we used random sampling, where the examples in each iteration were selected randomly.

To the best of our knowledge, there are no publicly-available datasets that provide true class probabilities for instances; hence there is no direct measure for the accuracy of CPEs. Instead, we use two indirect metrics proposed in other studies for CPEs [16]. The first metric is squared error, which is defined for an instance x_j, as $\sum_y (P_{true}(y|x_j) - P(y|x_j))^2$; where $P(y|x_j)$ is the predicted probability that x_j belongs to class y, and $P_{true}(y|x_j)$ is the true probability that x_j belongs to y. We compute the Mean Squared Error (MSE) as the mean of this squared error for each example in the test set. Since we only know the true class labels and not the probabilities, we define $P_{true}(y|x_j)$ to be 1 when the class of x_j is y and 0 otherwise. Given that we are comparing with

[3] J48 is the Weka [15] implementation of C4.5

this extreme distribution, squared error tends to favor classifiers that produce accurate classification, but with extreme probability estimates. Hence, we do not recommend using this metric by itself.

The second measure we employ is the area under the lift chart (AULC) [17], which is computed as follows. First, for each class k, we take the $\alpha\%$ of instances with the highest probability estimates for class k. r_α is defined to be the proportion of these instances actually belonging to class k; and r_{100} is the proportion of all test instances that are from class k. The lift $l(\alpha)$, is then computed as $\frac{r_\alpha}{r_{100}}$. The AULC_k is calculated by numeric integration of $l(\alpha)$ from 0 to 100 with a step-size of 5. The overall AULC is computed as the weighted-average of AULC_k for each k; where AULC_k is weighted by the prior class probability of k according to the training set. AULC is a measure of how good the probability estimates are for ranking examples correctly, but not how accurate the estimates are. However, in the absence of a direct measure, an examination of MSE and AULC in tandem provides a good indication of CPE accuracy. We also measured log-loss or cross-entropy, but these results were highly correlated with MSE, so we do not report them here.

To effectively summarize the comparison of two algorithms, we compute the percentage reduction in MSE of one over the other, averaged along the points of the learning curve. We consider the reduction in error to be *significant* if the difference in the errors of the two systems, averaged across the points on the learning curve, is determined to be statistically significant according to paired t-tests ($p < 0.05$). Similarly, we report the percentage *increase* in AULC.[4]

5.2 Results

The results of all our comparisons are presented in Tables 1-3. In each table we present two active learning methods compared to random sampling as well as to each other. We present the statistics *% MSE reduction* and *% AULC increase* averaged across the learning curves. All statistically significant results are presented in bold font. The bottom of each table presents the win/draw/loss (w/d/l) record; where a win or loss is only counted if the improved performance is determined to be significant as defined above.

5.3 Bootstrap-JS, Bootstrap-LV and Bootstrap-LV-EXT

We first examine the performance of BOOTSTRAP-JS for binary-class problems and compared it with that of BOOTSTRAP-LV and of random sampling. As shown in Table 1, BOOTSTRAP-JS often exhibits significant improvements over BOOTSTRAP-LV, or is otherwise comparable to BOOTSTRAP-LV. For all data sets, BOOTSTRAP-JS shows substantial improvements with respect to examples selected uniformly at random on both MSE and AULC. The effectiveness of BOOTSTRAP-JS can be clearly seen in Figure 1. (The plot shows the part of learning curve where the two active learners diverge in performance.)

In the absence of an active class probability estimation approach that can be applied to multi-class problems, we compare BOOTSTRAP-JS and BOOTSTRAP-LV-EXT with

[4] A larger AULC usually implies better probability estimates.

Table 1. BOOTSTRAP-JS versus BOOTSTRAP-LV on binary datasets

Data set	%MSE Reduction			%AULC Increase		
	LV vs. Random	JS vs. Random	JS vs. LV	LV vs. Random	JS vs. Random	JS vs. LV
breast-w	**14.92**	**14.81**	-0.12	**0.55**	**0.52**	-0.02
colic	**-1.45**	-0.04	**1.39**	**-0.95**	**-0.56**	**0.41**
credit-a	**2.1**	**3.98**	**1.92**	**-0.49**	-0.01	**0.48**
credit-g	-0.16	**0.77**	**0.93**	-0.01	**0.3**	**0.32**
diabetes	**1.01**	**1.75**	**0.75**	0.18	**0.58**	**0.4**
heart-c	**1.68**	0.29	**-1.43**	**0.57**	-0.08	**-0.64**
hepatitis	0.19	**2.64**	**2.43**	0.19	**1.03**	**0.84**
ion	**10.65**	**12.26**	**1.82**	**1.13**	**0.96**	-0.16
kr-vs-kp	**38.97**	**43**	**8.07**	**1.64**	**1.79**	**0.15**
sick	**19.97**	**20.84**	**1.03**	**0.62**	**0.41**	**-0.21**
sonar	**2.44**	**1.32**	**-1.17**	**0.58**	**0.74**	**0.16**
vote	**6.3**	**9.14**	**3.08**	**0.28**	**0.46**	**0.18**
w/d/l	9/2/1	10/2/0	9/1/2	7/3/2	9/2/1	8/2/2

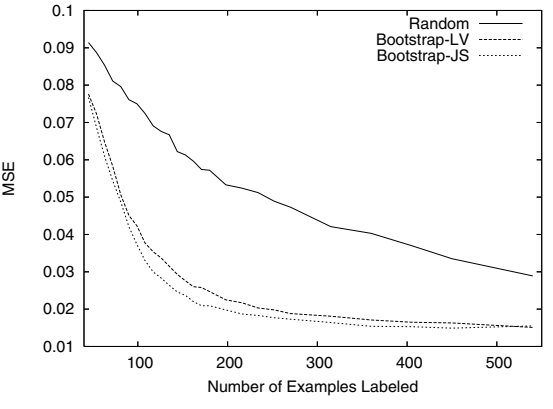

Fig. 1. Comparing different algorithms on *kr-vs-kp*

acquisitions of a representative set of examples selected uniformly at random. Table 2 presents results on multi-class datasets for BOOTSTRAP-JS and BOOTSTRAP-LV-EXT. Both active methods acquire particularly informative examples, such that for a given number of acquisitions, both methods produce significant reductions in error over random sampling. The two active methods perform comparably to each other for most data sets, and JSD performs slightly better in some domains. Because JSD successfully measures the uncertainty of the distribution estimation over all classes, we would recommend using BOOTSTRAP-JS for actively learning CPE models in multi-class domains.

5.4 ActiveDecorate: JSD Versus Margins

Table 3 shows the results of using JSD versus margins for ACTIVEDECORATE. In previous work, it was shown that ACTIVEDECORATE, with both these measures, performs very well on the task of active learning for classification. Our results here confirm that

Table 2. BOOTSTRAP-JS versus BOOTSTRAP-LV-EXT on multi-class datasets

Data set	% MSE Reduction			% AULC Increase		
	LV-Ext vs. Rand.	JS vs. Rand.	JS vs. LV-Ext	LV-Ext vs. Rand.	JS vs. Rand.	JS vs. LV-Ext
anneal	**12.27**	**13.06**	**0.89**	0.05	**0.5**	**0.45**
autos	**0.96**	**0.38**	**-0.58**	**1.51**	**0.83**	**-0.66**
balance-s	**1.39**	**0.92**	-0.48	**0.72**	**0.58**	-0.14
car	**7.21**	**6.93**	**-0.31**	**1.53**	**1.41**	**-0.12**
glass	**-0.55**	-0.19	**0.36**	**0.61**	**0.48**	-0.11
hypo	**46.62**	**46.41**	-0.9	**0.49**	**0.47**	-0.02
iris	**6.64**	**10.79**	**4.58**	**0.46**	**0.83**	**0.39**
nursery	**14.37**	**14.25**	-0.20	**0.44**	**0.42**	-0.01
optdigits	0.35	**0.71**	**0.35**	**0.9**	**1.13**	**0.23**
segment	**11.08**	**11.19**	0.08	**0.83**	**0.79**	-0.04
soybean	**1.5**	**0.78**	**-0.74**	**-0.46**	**0.4**	**0.87**
wine	**13.13**	**13.34**	0.36	**1.11**	**1.08**	-0.02
w/d/l	10/1/1	11/1/0	4/5/3	10/1/1	12/0/0	4/6/2

both measures are also effective for active learning for CPE. ACTIVEDECORATE using margins focuses on picking examples that reduce the uncertainty of the classification boundary. Since having better probability estimates usually improves accuracy, it is not surprising that a method focused on improving classification accuracy selects examples that may also improve CPE. However, using JSD directly focuses on reducing the uncertainty in probability estimates and hence performs much better on this task than margins. On the AULC metric both measures seem to perform comparably; however, on MSE, JSD shows clear and significant advantages over using margins. As noted above, one needs to analyze a combination of these metrics to effectively evaluate any active CPE method. Figure 2 presents the comparison of ACTIVEDECORATE with JSD versus margins on the AULC metric on *glass*. The two methods appear to be comparable, with JSD performing better earlier in the curve and margins performing better later. However, when the two methods are compared on the same dataset, using the MSE metric (Figure 3), we note that JSD outperforms margins throughout the learning curve. Based on the combination of these results, we may conclude that using JSD is more likely to produce accurate CPEs for this dataset. This example reinforces the need for examining multiple metrics.

5.5 ActiveDecorate-JS vs Bootstrap-JS

In addition to demonstrating the effectiveness of JSD, we also compare the two active CPE methods that use JSD. The comparison is made in two scenarios. In the *full dataset* scenario, the setting is the same as in previous experiments. In the *early stages* scenario, each algorithm is allowed to select 1 example at each iteration starting from 5 examples and going up to 20 examples. This characterizes the performance at the beginning of the learning curve. In the interest of space, we only present the win/draw/loss statistics (Table 4). For the *full dataset*, on the AULC metric, the methods perform comparably, but BOOTSTRAP-JS outperforms ACTIVEDECORATE-JS on MSE. However, for most datasets, ACTIVEDECORATE-JS shows significant advantages over BOOTSTRAP-JS in

Table 3. ACTIVEDECORATE-JS versus Margins

Data set	% MSE Reduction			% AULC Increase		
	Margin vs. Rand.	JS vs. Rand.	JS vs. Margin	Margin vs. Rand.	JS vs. Rand.	JS vs. Margin
breast-w	**9.32**	**23.91**	**12.73**	**0.29**	-0.50	**-0.79**
colic	**8.65**	**17.99**	**10.17**	**4**	**2.44**	**-1.47**
credit-a	**15.83**	**21.97**	**7.08**	**2.85**	**2.98**	0.07
credit-g	**7.06**	**8.91**	**2.02**	**6.98**	**7.79**	**0.75**
diabetes	**-3.11**	0.07	**2.9**	**4.98**	**0.84**	**-3.94**
heart-c	**4.66**	**6.3**	**1.72**	**1.54**	**0.53**	**-0.99**
hepatitis	**4.49**	**7.34**	**2.99**	**1.93**	**0.14**	**-1.95**
ion	**29.23**	**36.51**	**10.01**	**5.73**	**5.53**	-0.2
kr-vs-kp	**34**	**65.27**	**50.77**	**6.46**	**2.19**	**-3.99**
sick	**39.18**	**64.38**	**42.24**	**10.49**	**9.11**	**-1.24**
sonar	**9.3**	**9.31**	0.15	**5.84**	**5.37**	-0.41
vote	**12.15**	**45.79**	**38.12**	**0.81**	**-0.51**	**-1.31**
anneal	**45.51**	**63.8**	**32.1**	**7.62**	**11.14**	**3.27**
autos	**8.32**	**11.38**	**3.57**	**15.34**	**11.52**	**-3.34**
balance-s	**14.1**	**24.63**	**12.05**	**5.24**	**6.14**	**0.86**
car	**2.9**	**53.32**	**52.27**	**5.56**	**16.23**	**10.3**
glass	**7.62**	**12.31**	**5.02**	**8.62**	**10.51**	**1.82**
hypo	**31.37**	**89.87**	**86.34**	**4.03**	**4.7**	**0.65**
iris	**-1.32**	**34.32**	**32.7**	**-1.56**	**1.52**	**3.16**
nursery	**2.62**	**69.99**	**69.52**	**0.56**	**6.43**	**5.9**
optdigits	**32.56**	**39.8**	**10.67**	**19.38**	**17.79**	**-1.4**
segment	**56.95**	**71.12**	**27.27**	**6.11**	**6.85**	**0.71**
soybean	**15.82**	**21.84**	**7.42**	**21.1**	**34.35**	**10.89**
wine	**17.09**	**28.85**	**13.81**	**1.66**	**1.17**	**-0.5**
w/d/l	22/0/2	23/1/0	23/1/0	23/0/1	22/2/0	10/3/11

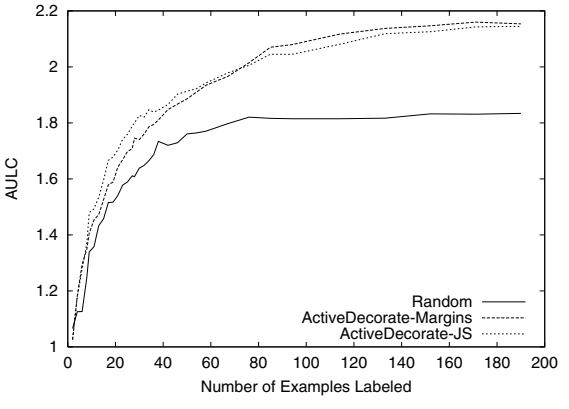

Fig. 2. Comparing AULC of different algorithms on *glass*

the *early stages*. These results could be explained by the fact that DECORATE (used byACTIVEDECORATE-JS) has a clear advantage over Bagging (used by BOOTSTRAP-JS) when training sets are small, as explained in [11].

Table 4. BOOTSTRAP-JS vs. ACTIVEDECORATE-JS: Win/Draw/Loss records

	% MSE Reduction	% AULC Increase
Full dataset	18/0/6	13/0/11
Early stages	8/2/14	2/5/17

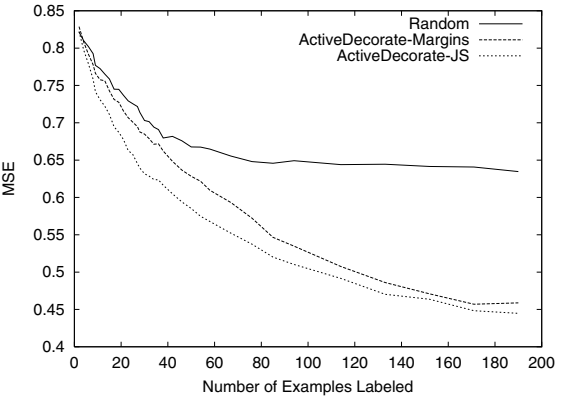

Fig. 3. Comparing MSE of different algorithms on *glass*

For DECORATE, we only specify the desired ensemble size; the ensembles formed could be smaller depending on the maximum number of classifiers it is permitted to explore. In our experiments, the desired size was set to 15 and a maximum of 50 classifiers were explored. On average DECORATE ensembles formed by ACTIVEDECORATE-JS are much smaller than those formed by Bagging in BOOTSTRAP-JS. Having larger ensembles generally increases classification accuracy [10] and may improve CPE. This may account for the weaker overall performance of ACTIVEDECORATE-JS to BOOTSTRAP-JS; and may be significantly improved by increasing the ensemble size.

6 Conclusions and Future Work

In this paper, we propose the use of Jensen-Shannon divergence as a measure of the utility of acquiring labeled examples for learning accurate class probability estimates. Extensive experiments have demonstrated that JSD effectively captures the uncertainty of class probability estimation and allows us to identify particularly informative examples that significantly improve the model's class distribution estimation. In particular, we show that, for binary-class problems, BOOTSTRAP-JS which employs JSD to acquire training examples is either comparable or significantly superior to BOOTSTRAP-LV, an existing active CPE learner for binary class problems. BOOTSTRAP-JS maintains its effectiveness for multi-class domains as well: it acquires informative examples which result in significantly more accurate models as compared to models induced from examples selected uniformly at random. We have also demonstrated that when JSD is used with ACTIVEDECORATE, an active learner for classification, it produces substantial improvements over using margins, which focuses on classification accuracy. Furthermore, our results indicate that, in general, BOOTSTRAP-JS with Bagged-PETs is a preferable

method for active CPE compared to ACTIVEDECORATE-JS. However, if one is concerned primarily with the early stages of learning, then ACTIVEDECORATE-JS has a significant advantage.

Our study uses standard metrics for evaluating CPE employed in existing research. However, we have shown that JSD is a good measure for selecting examples for improving CPE; and therefore it should also be a good measure for evaluating CPE. When the true class probabilities are known, we propose to also evaluate CPEs by computing the JSD between the estimated and the true class distributions.

Acknowledgments

This research was supported by DARPA grant HR0011-04-1-007.

References

1. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning **15** (1994) 201–221
2. Saar-Tsechansky, M., Provost, F.J.: Active learning for class probability estimation and ranking. In: Proc. of 17th Intl. Joint Conf. on Artificial Intelligence. (2001) 911–920
3. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, New York, NY (1991)
4. Melville, P., Mooney, R.J.: Diverse ensembles for active learning. In: Proc. of 21st Intl. Conf. on Machine Learning (ICML-2004), Banff, Canada (2004) 584–591
5. Dhillon, I., Mallela, S., Kumar, R.: Enhanced word clustering for hierarchical classification. In: Proc. of 8th ACM Intl. Conf. on Knowledge Discovery and Data Mining (2002)
6. McCallum, A., Nigam, K.: Employing EM and pool-based active learning for text classification. In: Proc. of 15th Intl. Conf. on Machine Learning (1998)
7. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: Proc. of 11th Intl. Conf. on Machine Learning, (1994) 148–156
8. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman and Hall, New York, NY (1993)
9. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proc. of the ACM Workshop on Computational Learning Theory, Pittsburgh, PA (1992)
10. Melville, P., Mooney, R.J.: Constructing diverse classifier ensembles using artificial training examples. In: Proc. of 18th Intl. Joint Conf. on Artificial Intelligence (IJCAI-2003), Acapulco, Mexico (2003) 505–510
11. Melville, P., Mooney, R.J.: Creating diversity in ensembles using artificial data. Journal of Information Fusion: Special Issue on Diversity in Multi Classifier Systems **6** (2004) 99–111
12. Saar-Tsechansky, M., Provost, F.: Active sampling for class probability estimation and ranking. Machine Learning **54** (2004) 153–178
13. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html (1998)
14. Provost, F., Domingos, P.: Tree induction for probability-based rankings. Machine Learning **52** (2003) 199–215
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)
16. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: Proc. of 18th Intl. Conf. on Machine Learning (2001)
17. Nielsen, R.D.: MOB-ESP and other improvements in probability estimation. In: Proc. of 20th Conf. on Uncertainty in Artificial Intelligence, (2004) 418–425