ESTIMATING CLASS MEMBERSHIP PROBABILITIES USING CLASSIFIER LEARNERS

Abstract. We present an algorithm, "Probing", which reduces learning an estimator of the probability of membership in a class to learning binary classifiers. The reduction comes with a theoretical guarantee: a small error rate for binary classification implies accurate estimation of class membership probabilities. We test this reduction on several datasets with several classifier learning algorithms and observe strong performance as compared to other common methods for obtaining class membership probability estimates from classifiers.

1. Introduction

Classifier learning is the problem of inducing a predictor for distinguishing between two (or a few) classes given a set of labeled training examples. There are many reasons for considering classifier learning as the most fundamental learning task. First, predicting one bit (or a few bits) is as simple as non-trivial prediction can be. Because of this simplicity, classifier learning is often more amenable to study than other learning problems. On the practical side, several fast learning algorithms (such as SVM [7], boosting [8] on decision trees [6], and neural networks [2]) exist which generally lead to classifiers with excellent predictive performance.

On the other hand, there are several real-world applications requiring probabilistic answers instead of discrete class labels. For example, in medical domains, doctors often prefer to make decisions based upon probabilistic predictions of the patient state, rather than being given a discrete chosen action for each patient. Even when the actual probability estimates are not required, a ranking of examples from the most likely member to the least likely member of a class may be desirable. This is the case, for example, in document categorization, where users might like to see a list of documents ranked by their relevance to a particular topic.

Probability estimates are also important when the classification outputs are not used in isolation but are combined with information from other components in a system. For example, in handwritten character recognition the outputs from the classifier are used as input to a high-level system which incorporates domain information, such as a language model. In this case, probabilities provide a standard way for summarizing the information from multiple sources into a single decision. When the components of the system are distributed, bandwidth constraints may make this summarization necessary as well as standard.

Our method for answering the needs from these real-world applications can be thought of as a reduction of class probability estimation to classifier learning. The reduction (which we name "Probing") satisfies strong optimality guarantees: good performance with respect to classification implies good performance with respect to class probability estimation. The essential observation underlying the Probing reduction is that probability estimates (in a Bayesian sense) can be extracted from preferences over bets at different odds ratios. The machinery that applies this observation to classifier learning algorithms is the Probing reduction.

2. Related Work

There are several existing approaches for obtaining class probability estimates from classifiers.

One common approach uses bagging [1]. Essentially, the learning algorithm is run on many training sets formed by sampling with replacement from the original dataset, and the "probability" that any sample x has label $y = 1$ is the proportion of learned classifiers that output 1 for x. The results of the bagging approach may have nothing to do with $\Pr_{(x,y)\ D}(y = 1|x)$, the probability according to the data generation process that $y = 1$. As has been observed before, the probability estimates obtained through bagging are measuring the instability of the classifier learning method over data perturbations [?] rather than $\Pr_{(x,y)\ D}(y = 1|x)$. Indeed, for larger datasets, bagging with certain learning algorithms might result in the same classifier on every run, causing the bagging probability to be always 0 or 1 even when the fundamental process is much less certain. In contrast, our Probing reduction has a simple guarantee—if the classifiers solve their associated classification problems optimally, then the reported probability is precisely $\Pr_{y \sim D|x}(y = 1)$. In addition, unlike bagging, Probing preserves independence in the original dataset[1], which is quite important for many learning algorithms (see [5] for further discussion).

Another approach for extracting probabilities from classifiers is to take an internal representation such as the margin of a support vector machine and transform it into a value $v \in [0, 1]$ to achieve calibration on the training set (for example, using a sigmoid function [4]). The Probing reduction is more general (since it applies to classifiers without an internal margin such as decision trees) and less ad-hoc. In particular, in the Probing reduction a small classification error rate necessarily implies good probability estimates over all distributions generating the data. No guarantee of this quality can be made for any transformation of a margin into a $[0, 1]$ interval, essentially because the margin is not a "sufficient statistic" for probability estimation. See also [3] for a discussion of sparsity vs probability estimation issues.

The third standard approach is to simply predict probabilities directly, typically with some form of learned probabilistic model. The results here can be viewed from this approach as (1) providing compatibility between non-probabilistic and probabilistic models and (2) enriching the set of probabilistic models with new and often better performing models.

The remainder of this paper presents the exact Probing reduction, analyzes it, and tests its empirical performance.

3. The Probing Reduction

Assume we have examples (x, y) drawn from an unknown distribution D with domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the feature space and $\mathcal{Y} = \{0, 1\}$ is the label space.

The Probing reduction is based upon the observation that a binary classifier which predicts 1 for an example x is implicitly predicting that the probability of class 1 given x is greater than the probability of class 0 given x, that is,

$$D(y = 1|x) > D(y = 0|x).$$

[1] Resampling with replacement provides samples from the correct distribution in a *dependent* manner.

Conversely, a classifier which predicts 0 for x suggests that

$$D(y=0|x) > D(y=1|x).$$

By weighting the training examples such that positive examples are w times more costly to classify incorrectly than negative examples, it is possible to learn classifiers which predict 1 if

$$D(y=1|x) > \frac{1}{1+w}$$

and 0 otherwise.

Theorem 3.1. *(Minimal Error Probing) Let $(x,y,i) \sim D'$ be $(x,y) \sim D$ with $i = wy + (1-y)$. Then*

$$\arg\min_{c:X\to Y} E_{(x,y,i)\sim D'}\ i\ I(c(x) \neq y) = I\left(D(y=1|x) > \frac{1}{1+w}\right)$$

This theorem states that a classifier which minimizes a importance weighted loss can be used to determine if the conditional distribution $D(y=1|x)$ is above or below a threshold for a given value of x, where the threshold depends on the relative weight given to positive and negative examples. Later, we state a theorem for the approximate case.

Proof. The translation proposition of [5] implies that

$$\frac{1}{Z} E_{(x,y,i)\sim D'}\ i\ I(c(x) \neq y) = E_{(x,y,i)\sim \hat{D}} I(c(x) \neq y)$$

where $\hat{D}(x,y,i) = \frac{i}{E_{(x,y,i)\sim D}\, i} D(x,y,i)$ and $Z = E_{(x,y,i)\sim D} i$.

Using this, we get:

$$\arg\min_{c:X\to Y} E_{(x,y,i)\sim D'}\ i\ I(c(x) \neq y) = \arg\min_{c:X\to Y} E_{(x,y)\sim \hat{D}} I(c(x) \neq y)$$

Noting that the optimal classifier always picks the most probable class and using the definition of $\hat{D}$, we get

$$= I\left(\hat{D}(y=1|x) > \hat{D}(y=0|x)\right) = I\left(\frac{w}{Z} D(y=1|x) > \frac{1}{Z} D(y=0|x)\right)$$

Using the fact that y can only take values 0 or 1 and applying algebra, we get

$$= I\left(wD(y=1|x) > 1 - D(y=1|x)\right) = I\left(D(y=1|x) > \frac{1}{1+w}\right).$$

□

Now, suppose that we learn a classifier c_w for every choice of w and each of these classifiers has a minimal loss. Then, for an input x, each of the classifiers answers whether $D(y=1|x)$ is greater than the threshold $\frac{1}{1+w}$. For $w=0$, we necessarily have $c_w(x) = 0$ because $D(y=1|x)$ cannot be greater than 1. As w grows, the predictions of the subsequent classifiers are monotone in w with one point w^* where the prediction changes from 0 to 1. Therefore, the Probing reduction reports:

$$D(y=1|x) = \frac{1}{1+w^*}. \tag{3.1}$$

Three issues remain:

Algorithm 1 **Probing-Learner** **(classifier learner A, dataset $S = (x, y)^m$, probability tests $\{p_1, ..., p_n\}$, where $1 \geq p_1 > \cdots > p_n > 0$)**

(1) Let $w_0 = 1$
(2) For p in $\{p_1, ..., p_n\}$
  (a) Let $w_1 = \frac{1-p}{p}$
  (b) Let $S_w = \{(x, y, w_y) : (x, y) \in S\}$
  (c) Let $c_w = A(S_w)$
(3) Return all c_w

Algorithm 2 **Probing-Predictor** **(set of weighted classifiers c_w, input x)**

(1) Let $n = \sum_w c_w(x)$
(2) Let w_n= the n-th largest weight.
(3) Return $p \in [\frac{1}{1+w_n}, \frac{1}{1+w_{n+1}}]$.

(1) Not all classifier learners automatically take the weight w as input. We address this by using the costing reduction [5] from importance weighted classification to binary classification, which applies to any classifier learner.
(2) Learning a classifier for every choice of w is too computationally intense. We resolve this by discretizing the set of w and reporting a probability in the discrete interval.
(3) Independently learned sub-optimal classifiers may not be consistent, meaning that the sequence of predictions for a given example is not necessarily monotonic. We simply sort the results of the classifiers to make the sequence monotonic (all zeros before all ones). This gives us the solution that is consistent with the largest possible number of classifiers.

The algorithms for learning and making predictions using the Probing reduction are reported in figures Algorithm 1 and Algorithm 2. The Probing-Learner algorithm takes as input a classifier learning algorithm, a set of training examples and a discrete set of possible probability values. For each probability value, it calculates the corresponding weight (obtained by inverting equation 3.1) and learns a weighted classifier using that weight. The Probing-Predictor algorithms takes as input the set of weighted classifiers and an input x. It sums the predictions of each classifier on x to determine the number of classifiers that predict 1 for x. It then picks a probability in the interval corresponding to the n-th and $(n+1)$-th largest weights (calculated using equation 3.1). This is equivalent to sorting the classifier predictions to obtain monotonicity and reporting the interval in which the prediction changes from 1 to 0.

Notice that the learning algorithm does not specify an exact set of discrete probabilities and that the prediction algorithm returns an interval for the predicted probability instead of an exact value. We leave these parameters indetermined because for different applications different discretizations might be preferable over others depending on how the accuracy of the predictions is assessed. Two common measures for quantifying the performance of a method for estimating class membership probabilities are cross entropy and mean squared error. Each one of them leads to a different heuristic for discretization:

(1) Cross entropy is given by

$$E_{(x,y)\sim D} I(y=1)\ln\frac{1}{\hat{p}(x)} + I(y=0)\ln\frac{1}{1-\hat{p}(x)}.$$

Therefore, when cross entropy is of interest, choosing the weights on an exponential schedule from some small value (about 10^{-3}) to some large value (say 10^3) may work well. The reported probability might be minimax optimal in the interval $[\frac{1}{1+w_n}, \frac{1}{1+w_{n+1}}]$ with respect to cross entropy.

(2) Mean squared error is given by

$$E_{(x,y)\sim D}[(y - D(y|x))^2].$$

Therefore, when squared loss is of interest, choosing the weights from a uniform grid over $[0,1]$ may work well.

In general, we can *adaptively* choose the weights given any particular probability-based performance metric. Given any set of tests $\{p_1 ... p_n\}$ we can augment the set with the next element p_i between two existing tests p_{i-1} and p_{i+1} which maximally improves the minimax optimal performance given the improved discretization. This approach works well in practice, particularly when the choice of interval is weighted by the distribution of the training set, that is, by the proportion of examples whose predictions change within the interval [2].

4. Analysis of the Probing Reduction

Here, we analyze the theoretical implications of the Probing reduction. For the following theorem, we use the discretized variant which chooses the probability tests from the uniform grid on $[0,1]$ and reports the mean value of the interval.

Theorem 4.1. *(Probing Error Transformation) If $D(1) = D(0) = 0.5$ and the classifiers learned under probing have average relative importance weighted loss*

$$\epsilon = E_{w\sim U(0,1)}\left[\frac{1}{N_w}E_{(x,y,i_w)\sim D'} i_w I(c(x)\neq y) - \frac{1}{N_w}E_{(x,y,i_w)\sim D'} i_w I(c^*(x)\neq y)\right],$$

where $c^*(x)$ is the Bayes optimal classifier for distribution D', then

$$E_{x,y\sim D}(D(y|x) - \hat{p}(x))^2 \leq \epsilon$$

where $\hat{p}(x)$ is the probing prediction.

There are two important points to notice about this theorem. First, this is a strong theorem in the sense that it depends on the *average relative* importance weighted loss. Using the average loss over w results in a more powerful theorem than using the maximal loss, because it is easier to obtain a set of classifiers which have small average loss than to bound the error for all classifiers. Using the relative loss as compared to the Bayes optimal classifier implies that the theorem applies even when the fundamental noise rate is large.

Second, this theorem is weaker than desired because it only applies to balanced distributions. A more complicated theorem (with weaker results) holds for unbalanced distributions. We note, however, that the degradation in the power of the reduction as the distribution becomes more unbalanced is not catastrophic.

Also note that we do not include the discretization error here. In general, discretization to intervals of size d adds at most $d/2$ to the squared error.

[2] In a transductive setting, using the test set distribution is an even better choice.

Proof. The proof works by analyzing the optimal strategy of a set of adversarial classifiers which suffer at most some relative importance weighted loss rate.

Based on this observation, we can say that adversarial classifiers all err in one direction (either for $p_w < D(1|x)$ or $p_w > D(1|x)$) so as to avoid cancellation of errors in the sorting phase of Probing.

First, we analyze the importance weighted error of a classifier. For any fixed choice of w we have:

$$\frac{1}{N_w}E_{(x,y,i)\sim D'}iI(c(x) \neq y) - \frac{1}{N_w}E_{(x,y,i)\sim D'}iI(c^*(x) \neq y)$$
$$= \frac{1}{N_w}E_{(x,y,i)\sim D'}i[I(c(x) \neq y) - I(c^*(x) \neq y)]$$
$$= \frac{1}{N_w}E_{x\sim D}E_{y\sim D|x}i_y[I(c(x) \neq y) - I(c^*(x) \neq y)]$$
$$= \frac{1}{N_w}E_{x\sim D}D(1|x)w[c^*(x) - c(x)] + D(0|x)[c(x) - c^*(x)]$$
$$= \frac{1}{N_w}E_{x\sim D}D(1|x)w[c^*(x) - c(x)] + (1 - D(1|x))[c(x) - c^*(x)]$$
$$= \frac{1}{N_w}E_{x\sim D}[c^*(x) - c(x)][(1 + w)D(1|x) - 1]$$

Rewriting with $p_w = 1/(1+w)$, $D(1|x) > p_w \Rightarrow c^*(x) = 1$, $D(1|x) < p_w \Rightarrow c^*(x) = 0$, we have:

$$= 2p_w E_{x\sim D}I(c^*(x) \neq c(x))\left|\frac{D(1|x)}{p_w} - 1\right|$$

So an error at any p_w implies a loss of:

$$2|D(1|x) - p_w|$$

Based on the above analysis, errors nearer to $D(1|x)$ are "cheaper" in importance weighted loss than errors far from $D(1|x)$.

To complete the analysis, we must consider how efficiently this induces loss in the probability estimation problem. The squared error loss of reporting $\hat{p}(x)$ is:

$$E_{x\sim D}(\hat{p}(x) - D(1|x))^2$$

Differentially, this loss is:

$$E_{x\sim D}2(\hat{p}(x) - D(1|x))$$

Consequently, we have that for every x, for every balanced D,

$$\frac{\text{change in squared error loss as } |\hat{p}(x) - D(1|x)| \text{ increases}}{\text{change in importance weighted loss } |\hat{p}(x) - D(1|x)| \text{ increases}} = 1$$

This implies that adversarial classifiers can induce maximal squared error loss with ϵ importance weighted loss by erring in the region $[D(1|x), \hat{p}(x)]$ or the region $[\hat{p}(x), D(1|x)]$ where $\hat{p}(x)$ is constrained in expectation by the pigeon hole principle so that

$$\epsilon = E_{x\sim D}\left|\int_0^{|\hat{p}(x)-D(1|x)|} 2z\,dz\right| = E_{x\sim D}(\hat{p}(x) - D(1|x))^2$$

where the last equality completes the proof. □

This basic theorem has implications on several metrics.

4.1. **The squared error metric for classification.** Since we typically do not consider situations where the class probabilities are known, it is important to realize that the above analysis applies to a performance metric that can be calculated entirely in terms of class labels.

The loss in the squared error metric for classification is defined as

$$E_{x,y\sim D}(y - \hat{p}(x))^2$$

Note that:

$$E_{x,y\sim D}(y - D(1|x))^2 + (\hat{p}(x) - D(1|x))^2$$
$$= E_{x\sim D} D(1|x)[(1-D(1|x))^2 + (\hat{p}(x) - D(1|x))^2] + (1-D(1|x))[D(1|x)^2 + (\hat{p}(x) - D(1|x))^2]$$
$$= E_{x\sim D} D(1|x)[1 - 2D(1|x)] + D(1|x)^2 + (\hat{p}(x) - D(1|x))^2$$
$$= E_{x\sim D} D(1|x) + \hat{p}(x)^2 - 2\hat{p}(x)D(1|x)$$
$$= E_{x\sim D}(\hat{p}(x) - D(1|x))^2 + D(1|x) - D(1|x)^2$$

Consequently, optimizing the squared error metric for classification implicitly optimizes the squared error in the probability estimate.

4.2. **The cross entropy metric for classification.** The loss in the cross entropy metric is $E_{x,y\sim D} I(y = 1) \ln \frac{1}{\hat{p}(x)} + I(y = 0) \ln \frac{1}{1-\hat{p}(x)}$. Cross entropy differs fundamentally from the squared error metric because it is (theoretically) unbounded. Consequently, the optimal strategy of an adversarial binary classifier is to err simultaneously in the same direction, producing an unbounded loss when the discretization goes to 0. For cross entropy we can only prove a much weaker theorem: minimal error implies $\hat{p}(x)$ is optimal up to the discretization.

In practice, this worst case behavior can be avoided. First, note that the asymptote to ∞ is very slow as $\hat{p}(x)$ approaches 0 or 1. Consequently, we can project $\hat{p}(x)$ into the nearest element in the interval $[0.001, 0.999]$ and incur a maximal loss of about $6.9\epsilon + 0.001$, while preserving almost all of the dynamic range.

4.3. **Implications for ordering metrics (for example, area under the ROC curve).** Very small errors in the ordering with respect to $[0, 1]$ can result in very large misorderings with respect to a subset of $[0, 1]$ In particular, suppose that the optimal answer is $\hat{p}(x) = 0.5$, but the answer we provide is $\hat{p}(x) = 0.49999$. If every other element in the ordering is in the interval $(0.49999, 0.5)$, then this small error results in a very large ordering error with respect to the set of elements.

5. Experimental Results

Here we present the results obtained by applying the Probing reduction to eight (binary) datasets from the UCI machine learning repository [9]. We use three different base classifier learners available within the machine learning tool Weka [10]: the J48 decision tree learner, the SMO support vector machine (SVM) learner and Naive Bayes. For choosing the weights, we use an adaptive discretization scheme that attempts to minimize cross entropy. For realizing the importance weighted classification, we use rejection sampling (proportionally to the weights) for the decision tree learner and give the weights directly to the learners for SVM and Naive Bayes, as done in Zadrozny et al. [5].

We are interested in comparing the performance of the Probing reduction with standard methods for obtaining probability estimates from these classifiers. Decision trees and Naive Bayes produce an internal score in the interval $[0, 1]$ for each

Dataset	J48 Decision Tree	Bagging J48	Probing J48
Adult	∞	0.49861	**0.43643**
Australian Credit	∞	0.44444	**0.44167**
Breast	∞	0.79324	**0.79114**
Diabetes	∞	0.80557	**0.78374**
Echocardiogram	1.21555	0.92399	**0.89694**
Hepatitis	0.73308	**0.65540**	0.66528
Liver	∞	**0.83225**	0.88789
TIC (Coil)	0.32546	0.32262	**0.30095**

TABLE 1. Cross entropy results using the J48 decision tree learner.

Dataset	J48 Decision Tree	Bagging J48	Probing J48
Adult	0.34704	0.33136	**0.30953**
Australian Credit	0.32747	0.30279	**0.29786**
Breast	0.43912	0.42644	**0.42617**
Diabetes	0.47429	0.42868	**0.42758**
Echocardiogram	0.53928	0.46991	**0.46622**
Hepatitis	0.39029	0.38311	**0.36801**
Liver	0.55933	**0.43859**	0.46118
TIC (Coil)	0.23656	0.23607	**0.23177**

TABLE 2. Squared error results using the J48 decision tree learner.

example that can be used as a class probability estimate, but they are known to be inaccurate []. For this reason, besides testing the performance of these scores, we have also tested well-known methods for obtaining better probability estimates from these classifiers: bagging for decision trees (100 classifiers) and calibration using a sigmoid function for Naive Bayes [?]. Since the SVM classifier does not produce probability estimates, we have tested only the standard method for obtaining probabilities from SVM (also available in Weka): calibration using a sigmoid function [4].

For datasets that have a standard training/test split in the UCI repository, we use the standard split. For the other datasets, we randomly split the data into a training set with 66% of the examples and a test set with 33% of the examples.

The results using the decision tree learner J48 are shown in tables 1 and 2. We report the cross entropy and the root mean squared error for each of the datasets using J48, Bagging J48 and Probing J48. The Probing reduction always outperforms a single J48 classifier and for most datasets also outperforms Bagging J48, on both metrics. A cross entropy of ∞ means that the classifier predicted probability one for an example whose label is zero (or vice-versa). This is often the case for the J48 classifier, showing that its internal scores are not reasonable class probability estimates for cross entropy loss.

The results using SVM are shown in tables 3 and 4. We report the cross entropy and the root mean squared error for each of the datasets using SVM with sigmoid calibration and Probing SVM. The Probing reduction outperforms the SVM with sigmoid calibration for most datasets on both metrics.

Dataset	Sigmoid SVM	Probing SVM
Adult	**0.47166**	0.47913
Australian Credit	0.48659	**0.47128**
Breast	0.83655	**0.80923**
Diabetes	**0.75768**	0.77505
Echocardiogram	0.89059	**0.83722**
Hepatitis	0.69502	**0.65979**
Liver	**0.88538**	0.89057
TIC (Coil)	0.32524	**0.31102**

TABLE 3. Cross entropy results using the SVM learner.

Dataset	Sigmoid SVM	Probing SVM
Adult	**0.32372**	0.32557
Australian Credit	0.32010	**0.31301**
Breast	0.43890	**0.43598**
Diabetes	**0.41222**	0.42388
Echocardiogram	0.46443	**0.44731**
Hepatitis	0.36795	**0.35820**
Liver	**0.45898**	0.46058
TIC (Coil)	0.23642	**0.23374**

TABLE 4. Squared error results using the SVM learner.

Dataset	Naive Bayes	Sigmoid NB	Probing NB
Adult	0.94912	0.52138	**0.46433**
Australian Credit	∞	**0.63666**	0.65604
Breast	0.97914	0.85110	**0.83000**
Diabetes	∞	0.86057	**0.82875**
Echocardiogram	0.90213	**0.81007**	0.91952
Hepatitis	1.41629	**0.73245**	0.95292
Liver	0.93987	**0.91716**	0.94970
TIC (Coil)	0.35723	**0.30500**	0.34816

TABLE 5. Cross entropy results using Naive Bayes (NB).

The results using Naive Bayes are shown in tables 5 and 6. We report the cross entropy and the root mean squared error for each of the datasets using Naive Bayes, Naive Bayes with sigmoid calibration and Probing Naive Bayes. The Probing reduction outperforms a single Naive Bayes classifier for most datasets on both metrics. Naive Bayes with sigmoid calibration, however, outperforms Probing Naive Bayes on most datasets. Part of the difficulty here is that the Naive Bayes classifier is not very accurate in comparison to the other classifiers, and so the probing reduction, which relies on that accuracy, is not optimal. On the other hand, the sigmoid approach has a seperate fitting stage which can repair bad estimates to some extent.

Dataset	Naive Bayes	Sigmoid NB	Probing NB
Adult	0.33522	0.33182	**0.31986**
Australian Credit	0.38197	**0.35772**	0.36304
Breast	0.47464	0.44680	**0.43190**
Diabetes	0.45923	0.44998	**0.43022**
Echocardiogram	0.44461	**0.43533**	0.45011
Hepatitis	0.43196	**0.40027**	0.40396
Liver	0.47264	**0.46117**	0.47898
TIC (Coil)	0.25355	**0.23298**	0.25353

TABLE 6. Squared error results using Naive Bayes (NB).

Dataset	Method (squared error)	Method (cross entropy)
Adult	Probing/J48	Probing/J48
Australian Credit	Probing/J48	Probing/J48
Breast	Probing/J48	Probing/J48
Diabetes	Sigmoid/SVM	Sigmoid/SVM
Echocardiogram	Sigmoid/NB	Sigmoid/NB
Hepatitis	Probing/SVM	Probing/J48
Liver	Bagging/J48	Bagging/J48
TIC (Coil)	Probing/J48	Probing/J48

TABLE 7. Best results over all algorithms and methods.

To get an idea of which algorithms are performing best, we have one last table consisting of the best test loss and method for each metric and dataset in table **??**. Probing generally performs well. [3]

Our experimental results show that Probing can be used to obtain accurate class probability estimates using a variety of classifier learners. Although classifier-specific methods for obtaining probabilities from classifiers can sometimes outperform Probing, it has the advantage of being a general purpose method that applies to any classifier learner (even ones that have not been created yet). Furthermore, because it has a solid theoretical foundation, it allows learning theory statements about classification to be transferred to class probability estimation.

References

[1]

[2] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient BackProp," in Neural Networks: Tricks of the trade, (G. Orr and Muller K., eds.), 1998.

[3] Peter L. Bartlett and Ambuj Tewari. Sparseness vs estimating conditional probabilities: Some asymptotic results. 2004. http://www.stat.berkeley.edu/~bartlett/papers/bt-secp-04.ps.gz

[4] J. Platt Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans, eds., pp. 61-74, MIT Press, (1999).

[5] Bianca Zadrozny, John Langford, and Naoki Abe Cost Sensitive Learning by Cost-Proportionate Example Weighting ICDM 2003

[6] Quinlan, J. R. (1993) C4.5: Programs for Machine Learning. San Mateo CA: Morgan Kaufmann.

[3]It's also interesting to note that the performance of the decision tree, J48. This may be partially due to 'overfitting on the UCI datasets'.

[7] B. Schölkopf and A. J. Smola. Learning with Kernels. MIT Press, 2002.
[8] Freund, Y. & Schapire R. E. (1997) A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, 55:1, 119-139.
[9] Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
[10] WEKA datamining software, Weka project, http://www.cs.waikato.ac.nz/~ml/weka/