Estimating Class Membership Probabilities using Classifier Learners

John Langford
Toyota Technological Institute at Chicago
Chicago, IL 60637
jl@tti-c.org

Bianca Zadrozny
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
zadrozny@us.ibm.com

Abstract

We present an algorithm, "Probing", which reduces learning an estimator of class probability membership 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 tested our reduction on several datasets with several classifier learning algorithms. The results show strong performance as compared to other common methods for obtaining class membership probability estimates from classifiers.

1 Introduction

Background. 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. 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 [12], boosting [6] on decision trees [13], and neural networks [8]) exist which generally lead to classifiers with excellent predictive performance.

The Problem. 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 language for summarizing 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.

Solution Sketch. Our method for answering the needs of 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.

Bagging. One common approach uses bagging [11]. Essentially, the learning algorithm is run on many training sets formed by sampling with replacement from the original dataset, and the estimated 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 $\Pr_{(x,y)\sim D}(y = 1|x)$, 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 [9] rather than $\Pr_{(x,y)\sim D}(y = 1|x)$. 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 $\Pr_{(x,y)\sim D}(y = 1|x)$. In addition, unlike bagging, Probing preserves independence in the original dataset[1], which is quite important for many learning algorithms (see [16] for further discussion).

Margin Sigmoid. 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 [10]). 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 [2] for a discussion of sparsity versus probability estimation issues.

Direct Prediction. 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.

Basic Observations. 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,

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

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.

Corollary 1 *(Minimal Error Probing) Let* $w_y = wy + (1 - y)$. *Then*

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

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

except on a set of measure 0.

This corollary follows from a more general theorem that we state and prove in section 4 (theorem 2). The corollary 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.

The Algorithm. 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:

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

Three issues remain:

1. Not all classifier learners automatically take the weight w as input. We address this by using the costing reduction [16] from importance weighted classification to binary classification, which applies to any classifier learner.

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

Algorithm 1 <u>**Probing-Learner**</u> **(classifier learner A, dataset $S = (x, y)^m$, number of iterations t)**

1. Let $I = \{[0, 1]\}$
2. For $i = 1$ to t
   (a) Let $[a, b]$ = minimax interval $\in I$
   (b) Let p = minimax optimal $p \in [a, b]$
   (c) Let $I = (I - \{[a, b]\}) \cup \{[a, p], [p, b]\}$
   (d) Let $w_1 = \frac{1-p}{p}$
   (e) Let $S_w = \{(x, y, w_y) : (x, y) \in S\}$
   (f) Let $c_w = A(S_w)$
3. Return all c_w

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

1. Let $n = \sum_w c_w(x)$
2. Let $[a, b]$ = the nth interval
3. Return minimax optimal $p \in [a, b]$.

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 consistent with the largest possible number of classifiers.

Learning. 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 some number of iterations. On each iteration, it refines the discretization of the interval with the largest potential gain from refinement (this is loss dependent, see below for details). For each chosen interval a minimax choice of probability and the corresponding weight (obtained by inverting equation 1) are calculated and a weighted classifier is learned.

Prediction. The Probing-Predictor algorithm 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 the minimax probability in the nth interval (counting from 0). This is equivalent to sorting the classifier predictions to obtain monotonicity and reporting a probability from the interval in which the prediction changes from 1 to 0.

Loss Details. Steps 2(a) and 2(b) in Probing-Learner are dependent on the probabilistic loss function of interest. Here we describe these steps for two of the most commonly used probabilistic loss functions: cross entropy and squared error. In order to determine the minimax interval and the minimax probability for these loss functions, we make the somewhat unjustified but convenient assumption that all classifiers are correct.

1. Cross entropy is given by
$$E_{(x,y)\sim D} I(y = 1) \ln \frac{1}{\hat{p}(x)} + I(y = 0) \ln \frac{1}{1 - \hat{p}(x)}.$$
The minimax optimal $\hat{p} \in [a, b]$ is given by:
$$\begin{aligned}&\min_{\hat{p}\in[a,b]} \max_{p\in[a,b]}[\text{loss on } \hat{p} - \text{loss on } p] \\ &= \min_{\hat{p}\in[a,b]} \max_{p\in[a,b]} \Big[p \ln \tfrac{1}{\hat{p}} + (1 - p) \ln \tfrac{1}{1-\hat{p}} \\ &\qquad - p \ln \tfrac{1}{p} - (1 - p) \ln \tfrac{1}{1-p}\Big]\end{aligned}$$
This implies that
$$\hat{p} = \frac{1}{1 + e^{\frac{H(a)-H(b)}{b-a}}},$$
where $H(a)$ is Shannon entropy (in nats) of a coin with bias a.
The minimax interval is the interval which creates the largest potential change in loss when refined on the training set. Suppose we let S_a be the set of training examples whose current probabilistic prediction is $\hat{p} \in [a, b]$. The chosen [2] interval is:
$$\arg\max_{[a,b]\in I} |S_a| \left[(1 - a) \ln \frac{1 - a}{1 - \hat{p}} + a \ln \frac{a}{\hat{p}}\right]$$
2. Mean squared error is given by
$$E_{(x,y)\sim D}[(y - D(y|x))^2].$$
The minimax optimal $\hat{p} \in [a, b]$ is given by:
$$\begin{aligned}&\min_{\hat{p}\in[a,b]} \max_{p\in[a,b]}[\text{loss on } \hat{p} - \text{loss on } p] \\ &= \min_{\hat{p}\in[a,b]} \max_{p\in[a,b]} \big[p(1 - \hat{p})^2 + (1 - p)\hat{p}^2 \\ &\qquad - p(1 - p)^2 - (1 - p)p^2\big]\end{aligned}$$
This implies that
$$\hat{p} = (b - a)/2.$$
The minimax interval is similarly given by
$$\arg\max_{[a,b]\in I} |S_a|(b - a).$$

[2] Note that the definition $\hat{p}$ implies this is equivalent to $\arg\max_{[a,b]\in I} |S_a| \left[(1 - b) \ln \frac{1-b}{1-\hat{p}} + b \ln \frac{b}{\hat{p}}\right]$

In a transductive setting, using the test set distribution to weight the minimax interval is a better than $|S_a|$.

Multiclass Case. For the multiclass case with $k > 2$ labels, we simply apply the binary method to $k-1$ binary problems defined by the mapping:

$$(x, y) \rightarrow (x, I(l = y))$$

for $k-1$ choices of l. This method gives us probability estimates $\hat{p}_1, ..., \hat{p}_{k-1}$ for $k-1$ classes, so we can estimate the probability of the last class according to $\hat{p}_k = 1 - \sum_{i=1}^{k-1} \hat{p}_i$.

4 Analysis of the Probing Reduction

The following theorem shows that the probing reduction is inherently robust against classification errors. For the theorem, let $w_y = wy + (1-y)$ and let the importance weighted loss of a classifier c under distribution D be

$$l_w(c) = \frac{1}{Z_w} E_{(x,y)\sim D} w_y I(c(x) \neq y)$$

where $Z_w = E_{(x,y)\sim D'}\, w_y$ is a normalization constant.

Theorem 2 (Probing Error Transformation) Let $w_y = wy + (1-y)$. If the classifiers c_w learned under Probing have average relative importance weighted loss

$$\epsilon = E_{\frac{1}{1+w}\sim U(0,1)} \left[l_w(c_w) - \min_c l_w(c) \right],$$

then

$$E_{x\sim D}(D(1|x) - \hat{p}(x))^2 \leq\; 2\max\{D(1), D(0)\}\epsilon$$

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

This is a strong theorem in the sense that it ties the error in the probability predictions to the *average relative* importance weighted loss of the classifiers. Using the average loss over w results in a more powerful statement than using the maximal loss, because it is easier to obtain a set of classifiers which have small average loss than to obtain a set of classifiers, all with a small loss. Using a loss that is relative to the loss of the Bayes optimal classifier means that the theorem applies even when the fundamental noise rate is large.

Note that when the proportion of positive and negative examples is balanced (D(1)=D(0)=0.5) the bound on the error in the probability predictions is exactly the average relative importance weighted loss,

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

Also note that we do not include the discretization error here. Discretization to intervals of size d adds at most $d + d^2/4$ to the expected squared error.

Proof: Let $c_w^* = \min_c l_w(c)$ and $p_w = 1/(1+w)$, then

$$\begin{aligned}
\epsilon &= E_{p_w\sim U(0,1)} \left[l_w(c_w) - l_D(c_w^*)\right] \\
&= E_{p_w}\left[\tfrac{1}{Z_w} E_{(x,y)\sim D}[w_y(I(c_w(x)\neq y) - I(c_w^*(x)\neq y))]\right] \\
&= E_{p_w}\left[\tfrac{1}{Z_w} E_{x\sim D} E_{y\sim D|x}[w_y(I(c_w(x)\neq y) \right. \\
&\qquad\qquad \left. - I(c_w^*(x)\neq y))]\right] \\
&= E_{p_w}\left[\tfrac{1}{Z_w} E_x[D(1|x)w(c_w^*(x) - c_w(x)) \right. \\
&\qquad\qquad \left. + D(0|x)(c_w(x) - c_w^*(x))]\right] \\
&= E_{p_w}\left[\tfrac{1}{Z_w} E_x[D(1|x)w(c_w^*(x) - c_w(x)) \right. \\
&\qquad\qquad \left. + (1 - D(1|x))(c_w(x) - c_w^*(x))]\right] \\
&= E_{p_w}\left[\tfrac{1}{Z_w} E_x[(c_w^*(x) - c_w(x))((1+w)D(1|x) - 1)]\right].
\end{aligned}$$

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

$$\epsilon = E_{p_w}\left[\frac{1}{p_w Z_w} E_x[I(c_w^*(x) \neq c_w(x))\, |D(1|x) - p_w|]\right].$$

Notice that for all w, $Z_w = D(0) + D(1)w$. Hence

$$p_w Z_w = \frac{1}{1+w} D(0) + \frac{w}{1+w} D(1) \leq \max\{D(1), D(0)\}.$$

Therefore, we have

$$\begin{aligned}
&E_{p_w} E_x[I(c_w^* \neq c_w(x))|D(1|x) - p_w|] \\
&\qquad \leq \max\{D(1), D(0)\}\epsilon,
\end{aligned}$$

which can be re-written as

$$\begin{aligned}
&E_{x\sim D} \int_0^1 I(c_w^* \neq c_w(x))|D(1|x) - p_w| d_{p_w} \\
&\qquad \leq \max\{D(1), D(0)\}\epsilon, \qquad (2)
\end{aligned}$$

Given a fixed budget of ϵ for binary errors, the scenario which maximizes probability error estimates has all classifiers erring in one direction (either for $p_w < D(1|x)$ or $p_w > D(1|x)$) because two errors in different directions are canceled in the sorting phase of Probing-Predictor. Furthermore, errors by classifiers nearer to $D(1|x)$ are preferred to errors far from $D(1|x)$, since the importance weighted loss payed by the adversary increases monotonically with distance from $D(1|x)$. Therefore, when the Probing prediction is $\hat{p}(x)$, all the classifiers c_w for which the corresponding p_w are between $\hat{p}(x)$ and the correct probability $D(1|x)$ give incorrect predictions whereas the other classifiers give correct predictions. As a result, the integrand of equation 2 is non-zero only for $\hat{p}(x) < p_w < D(1|x)$ (if the classifiers err in one direction) or $D(1|x) < p_w < \hat{p}(x)$ (if the classifiers err in the other direction). Thus, we can re-write equation 2 as

$$E_{x\sim D}\left| \int_{\hat{p}(x)}^{D(1|x)} |D(1|x) - p_w| d_{p_w} \right| \leq \max\{D(1), D(0)\}\epsilon.$$

By solving the integral, we get

$E_{x\sim D}(D(1|x) - \hat{p}(x))^2 \leq 2\max\{D(1), D(0)\}\epsilon.$ ∎

In this theorem, we measure the probabilistic loss using the true distribution $D(1|x)$. Because this distribution is unknown for most real-world problems, it is important to understand the implications of the theorem to metrics that only use the class labels. We consider these implications for three of these metrics in the following subsections.

4.1 The squared error metric for classification

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

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

Note that

$$\begin{aligned}
&E_{x,y\sim D}(y - D(1|x))^2 + (\hat{p}(x) - D(1|x))^2 \\
&= E_{x \sim D} D(1|x)[(1 - D(1|x))^2 + (\hat{p}(x) - D(1|x))^2 \\
&\qquad +(1 - D(1|x))[D(1|x)^2 + (\hat{p}(x) - D(1|x))^2] \\
&= E_x D(1|x)[1 - 2D(1|x)] + D(1|x)^2 + (\hat{p}(x) - D(1|x))^2 \\
&= E_x D(1|x) + \hat{p}(x)^2 - 2\hat{p}(x)D(1|x) \\
&= E_x(\hat{p}(x) - D(1|x))^2 + D(1|x) - D(1|x)^2 \\
&= E_x(\hat{p}(x) - D(1|x))^2 + C
\end{aligned}$$

where C is some problem dependent constant.

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 with all classifiers on one side of $D(1|x)$, 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 $0.001 + \epsilon \log \frac{1}{0.001} \simeq 6.9\epsilon + 0.001$ (when the classes are balanced), while preserving almost all of the dynamic range.

4.3 Implications for ordering metrics

Relative Ordering. In some situations, we may be interested only in the relative ordering or ranking of examples given by the class probability estimates. Unfortunately, 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.

AUC analysis. Area under the ROC curve (AUC) is one commonly used relative ordering metric. Suppose we have one example with label $y = 1$, $D(y = 1|x) = 1$ and Probing estimates $\hat{P}(y = 1|x) = 0$, and n examples with label $y = 0$, $D(Y = 1|x) = 1/n$ where Probing estimates $\hat{P}(y = 1|x) = 1/n$. In this setting, the AUC (as defined in [1]) is 0. Despite this terrible performance, the classifiers need err only once. This is less than any fixed importance weighted loss rate for sufficiently large n.

5 Experimental Results

Here we present the results obtained by applying the Probing reduction to fifteen (binary) datasets: eleven from the UCI machine learning repository [5] (Adult, Australian Credit, Breast, Diabetes, Echocardiogram, Hepatitis, Ionosphere, Kr-vs-kp, Liver, Mushroom, Sick), two from the UCI KDD archive [7] (KDD Cup 98 and COIL) and two from the 2004 KDD Cup (Biology and Physics).

We use three different base classifier learners available within the machine learning tool Weka [14]: the J48 decision tree learner, the SMO support vector machine (SVM) learner (linear kernel) and Naive Bayes. For choosing the weights, we use the Probing variant that attempts to optimize cross entropy for 100 iterations. 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. [16].

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 $[0, 1]$ that can be used as a class probability estimate, but they are known to be inaccurate [15]. For this reason, besides testing their performance, 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 [3]. 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 [10].

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.

We use three different metrics for assessing the accuracy of the probability estimates: root mean squared error (RMS), cross entropy (CXE) and area under the ROC curve (AUC). For more details about these metrics and an analysis of their characteristics, see [4]. The results on the test set using CXE, RMS and AUC are shown, respectively, in tables 1, 2 and 3. Note that for RMS and CXE the smaller the value the better the performance, while for AUC the larger the value the better the performance. Note also that AUC only measures how well the probability estimates rank the examples from the most probable to the least probable member of the class, while RMS and CXE also measure the calibration of the estimates.

Decision tree results. The Probing reduction outperforms a single J48 classifier for all datasets but Mushroom and for the majority of the datasets also outperforms Bagging J48, on the three 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. Bagging solves this problem by taking advantage of the instability of the learner over different training sets to average out extreme results. However, in cases where there is no instability (like the KDD Cup 98 dataset) bagging performs the same as the base classifiers. The same would be true if we apply bagging to other learners that are stable such as SVM and Naive Bayes.

SVM Results. The Probing reduction outperforms the SVM with sigmoid calibration for most datasets on all three metrics. The AUC metric results show a greater advantage of Probing over sigmoid calibration than the two other metrics. This is probably due to the fact that the sigmoid calibration maintains the same ranking as given by the SVM margins while Probing can give a better ranking based on the results of the different weighted classifiers.

Naive Bayes The Probing reduction outperforms a single Naive Bayes classifier for most datasets on the cross entropy and squared error metrics. On the AUC metric, however, a single Naive Bayes classifier usually performs slightly better than Probing Naive Bayes. This may be due to Naive Bayes estimates ranking well (although with terrible probability estimates), while Probing estimates have less resolution leading to more ties. The results also show that Naive Bayes with sigmoid calibration outperforms Probing Naive Bayes on most datasets in all three metrics. Naive Bayes is not a very accurate classifiers so the Probing reduction, which relies on that accuracy for different weight settings, is not optimal. On the other hand, the sigmoid approach has a separate fitting stage which can repair bad estimates to some extent. Furthermore, the sigmoid transformation maintains approximately the same ranking of examples as the original Naive Bayes classifier, leading to approximately the same AUC.

Dataset	RMS	CXE	AUC
Adult	Prob-J48	Prob-J48	Prob-J48
Australian	Prob-J48	Prob-J48	Prob-J48
Breast	Prob-J48	Prob-J48	Prob-NB
Biology	Sig-SVM	Sig-SVM	Prob-J48
COIL	Prob-J48	Prob-J48	Prob-J48
Diabetes	Sig-SVM	Sig-SVM	Prob-SVM
Echocardio	Sig-NB	Sig-NB	Sig-NB
Hepatitis	Bag-J48	Prob-SVM	Prob-NB
Ionosphere	Bag-J48	Bag-J48	Bag-J48
KDD Cup 98	Prob-J48	Prob-J48	Prob-J48
Kr-vs-kp	Bag-J48	Bag-J48	Bag-J48
Liver	Bag-J48	Bag-J48	Bag-J48
Mushroom	J48/SVM	J48/SVM	J48/SVM
Physics	Prob-J48	Prob-J48	Prob-J48
Sick	Bag-J48	Bag-J48	Prob-J48

Table 4: Best method for each dataset on each performance metric.

To get a better idea of which algorithms are performing best, we have one last table consisting of the method that led to best test performance for each metric and dataset in table 4. Probing wins in 23 out of 45 experiments against the other tested alternatives (J48, Naive Bayes, Sigmoid Naive-Bayes, Sigmoid-SVM, and Bagged Decision Tree). It is also interesting to note that the performance of the decision tree, J48, is very good. This may be due[3] to the fact that the other learners that we are using are linear (we have not tried different kernels for SVM).

Finally, to give an idea of the speed of convergence of Probing in practice, we plot the cross entropy performance (using J48) against the number of Probing iterations for 5 representative datasets in figure 1. Probing seems to converge to a reasonable probability estimate after 20 or so iterations.

6 Discussion

Probing is a general technique for converting any classifier learner into a class probability estimator. Theorem 2 shows that the probing reduction is well founded: good performance on the created classification problems implies good performance with respect to Probing probability estimates. Experimental results show that Probing achieves strong performance relative to a variety of other (typically more specialized) techniques using a variety of classifier learners. This combination of positive results is unmatched by any alternative method.

[3]It's also possible that the design of J48 (which inherits from C4.5) is overfit to the UCI datasets.

Dataset	J48	Bag-J48	Prob-J48	Sig-SVM	Prob-SVM	NB	Sig-NB	Prob-NB
Adult	∞	0.4986	**0.4364**	**0.4717**	0.4791	0.9491	0.5214	**0.4643**
Australian	∞	0.4444	**0.4418**	0.4866	**0.4713**	∞	**0.6367**	0.6560
Breast	∞	0.7932	**0.7911**	0.8366	**0.8092**	0.8398	**0.7527**	0.8300
Biology	∞	0.0179	**0.0170**	**0.0155**	0.0208	∞	0.0714	**0.0567**
COIL	0.3255	0.3226	**0.3010**	0.3252	**0.3110**	0.3572	**0.3050**	0.3482
Diabetes	∞	0.8056	**0.7837**	**0.7577**	0.7751	∞	0.8606	**0.7803**
Echocardio	1.2156	0.9240	**0.8969**	0.8906	**0.8372**	0.9021	**0.8101**	0.8516
Hepatitis	0.7331	**0.6554**	0.6653	0.6950	**0.6598**	1.4163	**0.7325**	0.9529
Ionosphere	∞	**0.2100**	0.2868	0.4183	**0.4160**	∞	0.3526	**0.3035**
KDD Cup 98	0.2888	0.2888	**0.2829**	0.2851	**0.2847**	∞	**0.2850**	0.3511
Kr-vs-kp	∞	**0.0263**	0.0685	0.1726	**0.1281**	0.4163	**0.4057**	0.4178
Liver	∞	**0.8323**	0.8879	**0.8854**	0.8906	0.9399	**0.9172**	0.9497
Mushroom	**0.0000**	0.0001	0.0056	**0.0000**	0.0008	0.1907	**0.1664**	0.2975
Physics	∞	0.7627	**0.7620**	**0.8016**	0.8052	∞	**0.8802**	0.9329
Sick	∞	**0.0700**	0.0856	**0.1708**	0.2025	0.2190	0.1899	**0.1833**

Table 1: Cross entropy results. The best results for each classifier (J48, SVM and Naive Bayes) are in bold.

Dataset	J48	Bag-J48	Prob-J48	Sig-SVM	Prob-SVM	NB	Sig-NB	Prob-NB
Adult	0.3470	0.3314	**0.3096**	**0.3237**	0.3256	0.3352	0.3318	**0.3199**
Australian	0.3275	0.3028	**0.2979**	0.32010	**0.3130**	0.3820	**0.3577**	0.3630
Biology	0.0609	**0.0480**	0.0484	**0.0465**	0.0524	0.0754	0.0940	**0.0567**
Breast	0.4391	0.4264	**0.4262**	0.4389	**0.4360**	0.4329	**0.4146**	0.4319
COIL	0.2365	0.2360	**0.2317**	0.2364	**0.2337**	0.2536	**0.2330**	0.2535
Diabetes	0.4743	0.4287	**0.4279**	**0.4122**	0.4239	0.4592	0.4500	**0.4293**
Echocardio	0.5393	0.4699	**0.4662**	0.4644	**0.4473**	0.4446	**0.4353**	0.4486
Hepatitis	0.3903	0.3831	**0.3680**	0.3680	**0.3582**	0.4320	**0.4003**	0.4040
Ionosphere	0.3033	**0.2014**	0.2146	0.2945	**0.2835**	0.2588	0.2511	**0.2374**
KDD Cup 98	0.2191	0.2191	**0.2181**	0.2186	**0.2184**	0.2334	**0.2185**	0.2470
Kr-vs-kp	0.0930	**0.0653**	0.0913	0.1788	**0.1604**	0.2997	**0.2948**	0.3001
Liver	0.5593	**0.4386**	0.4612	**0.4590**	0.4606	0.4726	**0.4612**	0.4790
Mushroom	**0.0000**	0.0006	0.0115	**0.0000**	0.0030	0.1817	**0.1699**	0.2014
Physics	0.5285	0.4245	**0.4236**	0.4348	**0.4344**	0.4900	0.4900	**0.4768**
Sick	0.1247	**0.1155**	0.1192	**0.1752**	0.2048	0.1991	**0.1866**	0.1897

Table 2: Squared error results. The best results for each classifier (J48, SVM and Naive Bayes) are in bold.

Dataset	J48	Bag-J48	Prob-J48	Sig-SVM	Prob-SVM	NB	Sig-NB	Prob-NB
Adult	0.8520	0.8910	**0.9124**	0.9006	**0.9012**	**0.9062**	**0.9062**	0.9049
Australian	0.8996	0.9373	**0.9441**	0.9121	**0.9238**	0.9200	**0.9201**	0.9172
Biology	0.8854	0.9755	**0.9873**	**0.9865**	0.9821	**0.9567**	0.9215	0.9556
Breast	0.6060	0.6020	**0.6331**	**0.6797**	0.6594	0.7226	0.7226	**0.7243**
COIL	0.5000	0.6412	**0.7197**	0.5419	**0.6817**	0.7086	0.7086	**0.7174**
Diabetes	0.7152	0.7955	**0.7978**	0.8166	**0.8241**	0.7785	0.7785	**0.8052**
Echocardio	0.5425	0.6437	**0.6448**	0.7035	**0.7425**	**0.7586**	**0.7586**	0.7528
Hepatitis	0.6540	**0.6698**	0.6619	**0.7492**	0.7476	0.7429	0.7429	**0.7571**
Ionosphere	0.8946	**0.9807**	0.9661	0.9262	**0.9275**	0.9599	**0.9602**	0.9567
KDD Cup 98	0.5000	0.5002	**0.6157**	0.5977	**0.6017**	**0.6064**	**0.6064**	0.6038
Kr-vs-kp	0.9979	**0.9997**	0.9994	0.9915	**0.9953**	**0.9534**	**0.9534**	0.9532
Liver	0.6067	**0.7537**	0.7235	0.7108	**0.7140**	**0.7130**	**0.7130**	0.7057
Mushroom	**1.0000**	**1.0000**	**1.000**	**1.0000**	**1.0000**	**0.9979**	**0.9979**	0.9898
Physics	0.6736	0.8026	**0.8034**	0.7845	**0.7885**	**0.7379**	**0.7379**	0.7291
Sick	0.9651	0.9850	**0.9889**	**0.9462**	0.9397	0.9491	0.9491	**0.9515**

Table 3: Area under the ROC results. The best results for each classifier (J48, SVM and Naive Bayes) are in bold.

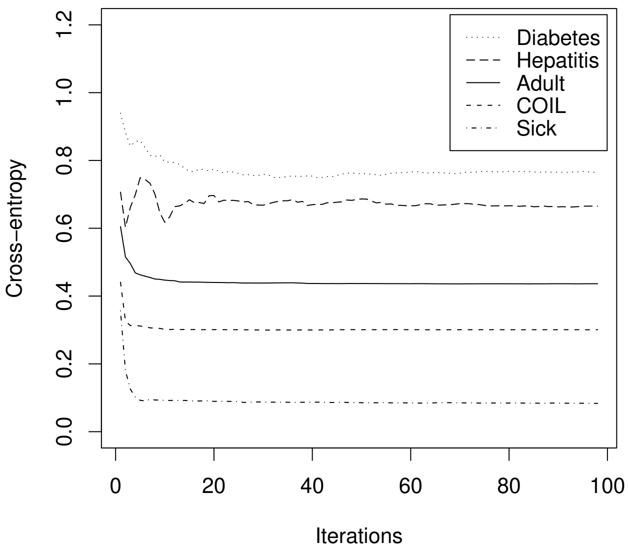

Figure 1: Cross entropy vs. the number of Probing iterations for 5 representative datasets (using J48).

In future work, we would like to validate the method on multiclass problems as outlined in section 3. Also, further studying the behavior of different discretization schemes for choosing the weights could lead to enhancements of the Probing estimates for different performance criteria.

References

[1] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled and D. Roth (2004). Generalization bounds for the Area under an ROC curve. Technical report UIUCDCS-R-2004-2433, Computer Science Department, University of Illinois at Urbana-Champaign.

[2] P. Bartlett and A. Tewari (2004). Sparseness versus estimating conditional probabilities: Some asymptotic results. In *Proceedings of the 17th Annual Conference on Learning Theory* (COLT-2004), 564-578. Springer-Verlag.

[3] P. Bennett (2000). Assessing the Calibration of Naive Bayes' Posterior Estimates (2000). Technical Report CMU-CS-00-155, Computer Science Department, Carnegie Mellon University.

[4] R. Caruana, A. Niculescu-Mizil (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining* (KDD-2004), 69-78. ACM Press.

[5] C. L. Blake and C. J. Merz (1998). UCI Repository of machine learning databases [`http://www.ics.uci.edu/~mlearn/MLRepository`]. Irvine, CA: University of California, Department of Information and Computer Science.

[6] Y. Freund and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of Computer and System Sciences*, 55:1, 119-139.

[7] S. Hettich and S. D. Bay (1999). The UCI KDD Archive [`http://kdd.ics.uci.edu`]. Irvine, CA: University of California, Department of Information and Computer Science.

[8] Y. LeCun, L. Bottou, G. Orr, and K. Muller (1998). Efficient BackProp. In G. Orr and K. Muller (eds.) *Neural Networks: Tricks of the trade.* Springer-Verlag.

[9] D. Margineantu (2000). On class probability estimates and cost-sensitive evaluation of classifiers. In *Workshop on Cost-Sensitive Learning, The Seventeenth International Conference on Machine Learning* (ICML-2000).

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

[11] F. Provost and P. Domingos (2003). Tree Induction for Probability-based Rankings. In *Machine Learning*, 52:3, 199-216. Kluwer.

[12] B. Schölkopf and A. J. Smola (2002). *Learning with Kernels.* MIT Press.

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

[14] I. H. Witten and E. Frank (1999). *Data mining: practical machine learning tools and techniques with Java implementations* [`http://www.cs.waikato.ac.nz/ml/weka/`]. Morgan Kaufmann.

[15] B. Zadrozny and C. Elkan (2001). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning* (ICML-01), 609-616. Morgan Kaufmann.

[16] B. Zadrozny, J. Langford, and N. Abe (2003). Cost Sensitive Learning by Cost-Proportionate Example Weighting. In *Proceedings of the 2003 IEEE International Conference on Data Mining* (ICDM-03), 435-442. IEEE Press.