EXTRACTION OF PROBABILITY FROM CLASSIFIERS

Abstract. We present an algorithm, "Probing", which reduces learning the probability of a class to binary classification. The reduction comes with a theoretical guarantee: a small error rate for binary classification implies accurate probability estimates. We test these reductions on several datasets with several classification algorithms and observe strong performance.

1. Introduction

Classification is the problem of predicting between two (or a few) classes (or actions) given data. The classification problem is motivated on several grounds:

(1) Classification is the simplest prediction problem. (Predicting one bit is as simple as nontrivial prediction can be.)
(2) On the theoretical side, classification is often more amenable to study than other prediction problems.
(3) On the practical side, several fast classification algorithms exist providing excellent performance.

On the other hand, there are several real-world applications requiring probabilistic (rather than class label or action choice) answers:

(1) Some applications insist on the reporting of probabilities rather than predicted classes (or actions). For example, in medical domains, doctors often prefer to make decisions based upon probabilistic predictions of patient state.
(2) In a distributed system, it may be desirable to minimize communication. By reporting the probability[1] of an event rather than all sensory information used to derive the probability, we can radically reduce the amount of communication required.
(3) In an engineered multielement learning system, components must be compatible. In particular, sometimes one of these components requires probabilistic input from another component.

Our method for answering these real-world applications can be thought of as a reduction of probability estimation to classification. The reduction (which we name "Probing") satisfies strong optimality guarantees: good performance with respect to classification results in good performance with respect to 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 machine applying this observation to classification algorithms is the Probing reduction.

There are several alternative approaches to reporting probabilistic predictions.

[1]Note that a single bit classification reduces this even further. However, there is typically some overhead cost of communicating any bits, so communicating a few rather than 1 has a similar cost. Also, reducing to just one bit often loses too much information.

One common approach uses "bagging" [1]. Essentially, the learning algorithm is run on many datasets 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. The results of the bagging approach may have nothing to do with $\Pr_{y\sim D|x}(y = 1)$, the probability according to the data generation process that $y = 1$. Indeed, for larger datasets bagging with certain 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, the reduction "probing" 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 [1], Probing preserves independence in the original dataset[2], which is quite important for many learning algorithms (see [3] for further discussion).

Another approach for extracting probabilities is to take an internal representation such as the margin of a support vector machine [2] and transform it into a value $v \in [0, 1]$ to achieve calibration on the training set. 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.

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 nonprobabilistic and probabilistic models (2) enriching the set of probabilistic models with new often better performing models.

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

2. The Probing Reduction

The Probing reduction is based upon the observation that a binary classifier which predicts 1 is implicitly predicting $\Pr_{y\sim D|x}(y = 1) > 0.5$. Furthermore, by creating importance weighted classification problems where the cost of a "false negative" is w and a "false positive" is 1, we can learn classifiers which differentiate $\Pr_{y\sim D|x}(y = 1) > \frac{1}{1+w}$. Suppose, that we learn a classifier for every choice of w and each of these classifiers has a minimal loss. Then, on a new input x, the predictions of these classifiers are monotone in w with one point w^* where the predictions change from 0 to 1. The Probing reduction reports:

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

Three issues remain:

(1) Classifiers naively do not take the weight w as input. We address this by using the costing reduction [3] from importance weighted classification to binary classification.

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

Algorithm 1 **Probing (Importance Weighted Classification Algorithm A, Dataset $S = (x,y)^m$, probability tests $\{p_1, ..., p_n\}$)**

(1) w_0
(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-Prediction (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 $p \in [\frac{1}{1+w_n}, \frac{1}{1+w_{n+1}}]$.

(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 classifiers may not be consistent. We simply sort the results of the classification to make them consistent.

The full algorithm is reported in figures 1 and 2. There is a slight indeterminancy in the returned value and the exact algorithm due to the issue of discretization. This issue is best handled in an application specific manner.

(1) 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) 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, particularly when the choice of interval is weighted by the distribution of the train set[3].

3. 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 3.1. *(Probing Error Transformation) If the learned classifiers have error rate ϵ above the minimal error rate possible, then*

$$E_{x,y\sim D}|P_{D|x}(y) - \hat{p}(x)| \leq \epsilon + \frac{d}{2}$$

[3]In a transductive setting, a test set is an even better choice.

where $\hat{p}(x)$ is the prediction of the reduction and d is the discretization.

Proof. (sketch, for the moment) After we renormalize the importance weighted problems, the cost of misclassification varies between 1 $\frac{1}{p_y}$ where p_y is the probability that D produces label y (aka the "prior"). If we assume (as in the worst case) that the cost is c, then the optimal strategy for an adversarial binary classifier is to always err in the same direction. Each (binary) error increases the $l_1$loss by the discretization d. Since there are $\frac{1}{d}$ classifiers and each can err an ϵ fraction of the time, the l_1 loss is at most $\epsilon + \frac{d}{2}$. □

This basic theorem has implications on several metrics.

3.1. **The Squared Error metric.** The loss in the squared error metric is defined as $E_{x,y\sim D}(P_{D|x}(y) - \hat{p}(x))^2$. The optimal strategy of an adversarial binary classifier is to err simultaneously, all in one direction. Consequently, an ϵ fraction of errors translates into an ϵ increase in the squared error (due to the case where $P_{D|x}(y) = 0$ and $\hat{p}(x) = 1$).

3.2. **The Cross Entropy metric.** The loss in the cross entropy metric is $E_{x,y\sim D} I(y = 1) \ln \frac{1}{\hat{p}(x)} + I(y = 0) \ln \frac{1}{1-\hat{p}(x)}$. Log loss 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, produced 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.

3.3. **Implications for ordering metrics (for example, area under the ROC curve).** The l_1 loss discussed in the main theorem can be thought of as a normalized sort distance from the actual to the optimal answer on a set of elements covering $[0, 1]$. Naturally, an optimal ordering on this metric translates into an optimal ordering over any subset of the interval$[0, 1]$.

However, 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.

4. Experimental Results

References

[1] Bagging probabilities
[2] Platt
[3] Costing