A Lazy Man's Approach to Benchmarking: Semisupervised Classifier Evaluation and Recalibration

Peter Welinder[*‡] Max Welling[†] Pietro Perona[‡*]
peter@welinder.se welling@cs.amsterdamuniversity.nl perona@caltech.edu
[*]Dropbox, Inc. [†]University of Amsterdam [‡]California Institute of Technology

Abstract

How many labeled examples are needed to estimate a classifier's performance on a new dataset? We study the case where data is plentiful, but labels are expensive. We show that by making a few reasonable assumptions on the structure of the data, it is possible to estimate performance curves, with confidence bounds, using a small number of ground truth labels. Our approach, which we call Semisupervised Performance Evaluation (SPE), is based on a generative model for the classifier's confidence scores. In addition to estimating the performance of classifiers on new datasets, SPE can be used to recalibrate a classifier by re-estimating the class-conditional confidence distributions.

1. Introduction

Training and testing on one image set is no guarantee of good performance on another [7, 13]. Consider an urban planner who downloads software for detecting pedestrians with the goal of counting pedestrians in the city center. The pedestrian detector was laboriously trained by a research group who labeled thousands of training and validation examples and publishes good experimental results (see e.g. [5]). Should the urban planner trust the published performance figures and assume that the detector will perform equally well on her images? Her images are mostly taken in an urban environment, while the authors of the detector used vacation photographs for training their system. Perhaps the detector is useless on images of urban scenes (Figure 1).

In order to be sure, the planner needs to compute precision and recall on her dataset. This requires labeling by hand a large number of her images, i.e. doing the detector's work by hand. What was then the point of obtaining a trained detector in the first place? What are the planner's options? Is it possible at all to obtain reliable bounds on the performance of a detector / classifier without relabeling a new dataset?

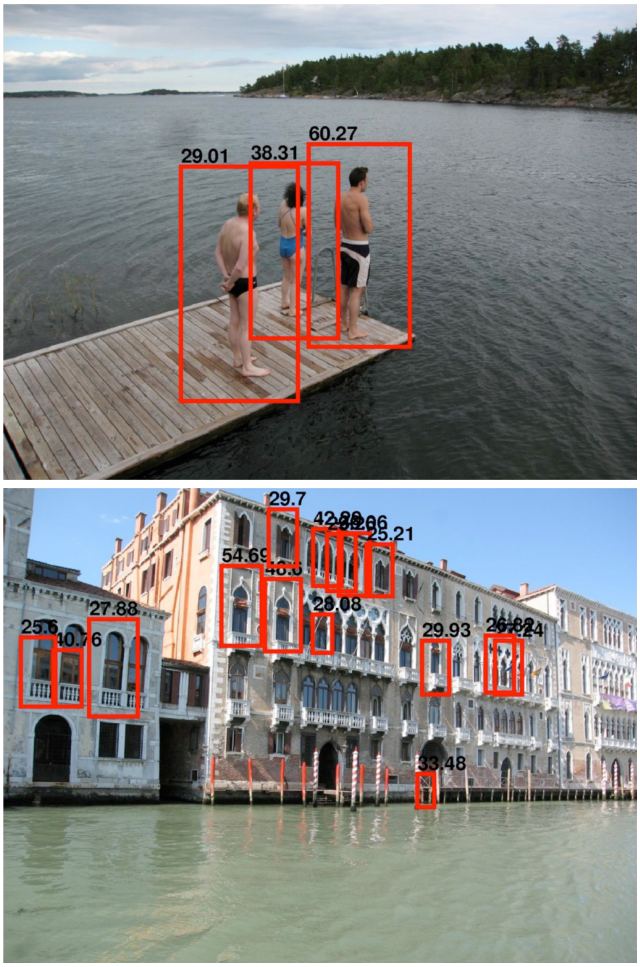


Figure 1. A pedestrian detector trained on vacation images (the INRIA dataset [5]) performs well on images taken in natural environments (top), and fails miserably on images taken in an urban environment (bottom). Can we estimate the performance of a pre-trained classifier / detector on a novel data set? Can we do so without expensive detailed labeling of a new ground truth dataset? Can we get reliable error bars on those estimates?

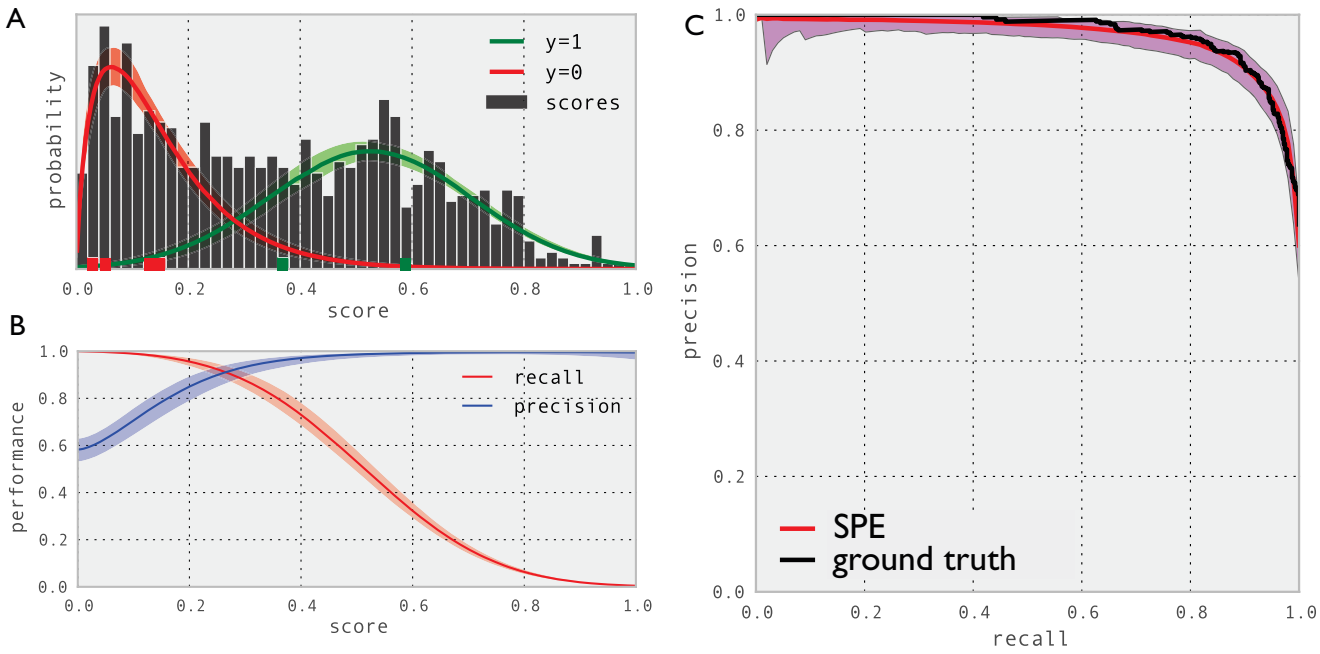


Figure 2. Estimating detector performance with 10 labels known. **A**: Histogram of classifier scores s_i obtained by running the "ChnFtrs" detector [7] on the INRIA dataset [5]. The red and green curves show the Gamma-Normal mixture model fitting the histogrammed scores with highest likelihood. The scores are all unlabeled, apart from 10, selected at random, which have labels. The shaded bands indicate the 90% probability bands around the model. The red and green bars show the labels of the 10 randomly sampled labels (by chance, the scores for some of the samples are close to each other, thus only 6 bars are shown; the height of the bars has no meaning). **B**: Precision and recall curves computed from the mixture model in A. **C**: In black, precision-recall curve computed after all items have been labeled. In red, precision-recall curve estimated using SPE from only 10 labeled examples (with 90% confidence interval shown as the magenta band). See Section 2 for a discussion.

We propose a method for achieving minimally supervised evaluation of classifiers, requiring as few as 10 labels to accurately estimate classifier performance. Our method is based on a generative Bayesian model for the confidence scores produced by the classifier, borrowing from the literature on semisupervised learning [16, 20, 21]. We show how to use the model to re-calibrate classifiers to new datasets by choosing thresholds to satisfy performance constraints with high likelihood. An additional contribution is a fast approximate inference method for doing inference in our model.

2. Modeling the classifier score

Let us start with a set of N data items, $(x_i, y_i) \in \mathcal{R}^D \times \{0, 1\}$, drawn from some unknown distribution $p(x, y)$ and indexed by $i \in \{1, \ldots, N\}$. Suppose that a classifier, $\bar{h}(x_i; \tau) = [h(x_i) > \tau]$, where τ is some scalar threshold, has been used to classify all data items into two classes, $\hat{y}_i \in \{0, 1\}$. While the "ground truth" labels y_i are assumed to be unknown, initially, we do have access to all the "scores," $s_i = h(x_i)$, computed by the classifier. From this point onwards, we forget about the data vectors x_i and concentrate solely on the scores and labels, $(s_i, y_i) \in \mathcal{R} \times \{0, 1\}$.

The key assumption in this paper is that the list of scores $S = (s_1, \ldots, s_N)$ and the unknown labels $Y = (y_1, \ldots, y_N)$ can be modeled by a two-component mixture model $p(S, Y \mid \theta)$, parameterized by θ, where the class-conditionals are standard parametric distributions. We show in Section 4.2 that this is a reasonable assumption for many datasets.

Suppose that we can ask an expert (the "oracle") to provide the true label y_i for any data item. This is an expensive operation and our goal is to ask the oracle for as few labels as possible. The set of items that have been labeled by the oracle at time t is denoted by $\mathcal{L}_t$ and its complement, the set of items for which the ground truth is unknown, is denoted $\mathcal{U}_t$. This setting is similar to semisupervised learning [20, 21]. By estimating $p(S, Y \mid \theta)$, we will improve our estimate of the performance of $\bar{h}$ when $|\mathcal{L}_t| \ll N$.

Consider first the fully supervised case, i.e. where all labels y_i are known. Let the scores s_i be i.i.d. according to the two mixture models. If the all labels are known, and we assume independent observations, the likelihood of the data is given by,

$$p(S, Y \mid \theta) = \prod_{i:y_i=0} (1 - \pi) p_0(s_i \mid \theta_0) \prod_{i:y_i=1} \pi p_1(s_i \mid \theta_1), \quad (1)$$

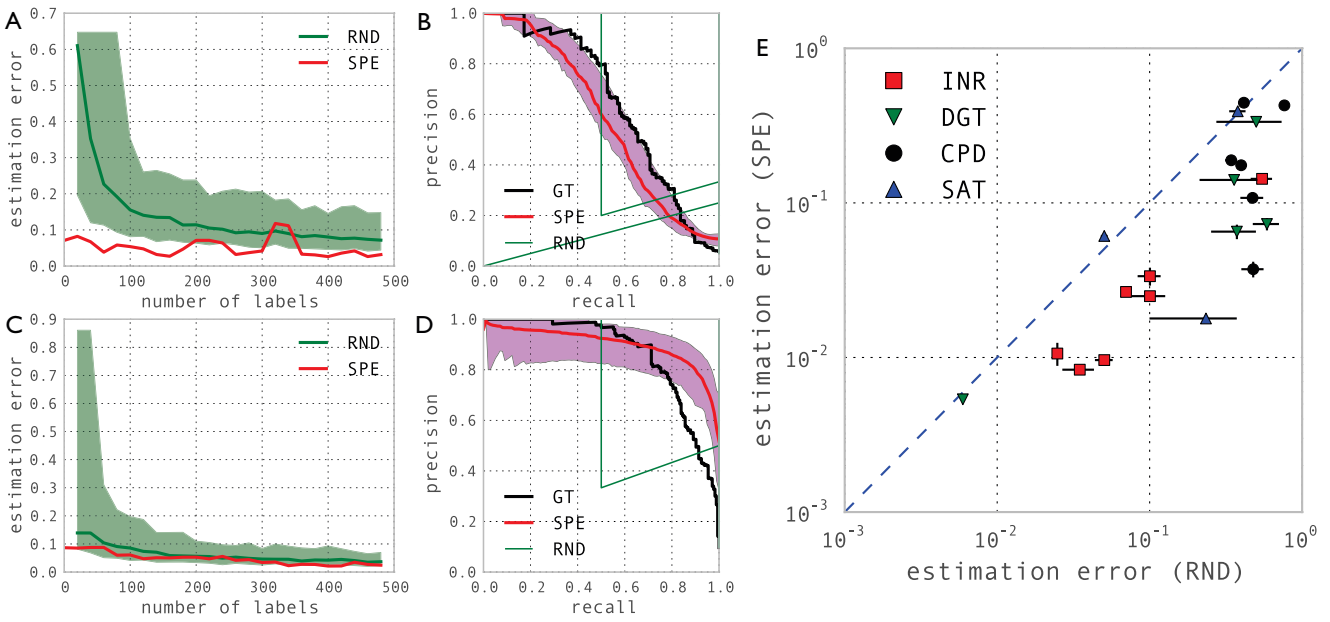


Figure 3. Applying SPE to different datasets. **A**: Estimation error, as measured by the area between the true and predicted precision-recall curves, versus the number of labels sampled, for the ChnFtrs detector on the CPD dataset. The red curve is SPE and the green curve shows the median error of the naive method (RND). The green band show the 90% quantiles of the naive method. **B**: The performance curve estimated using SPE (red) with 90% confidence intervals (magenta) with 20 known labels. The ground truth performance with all label known is shown as a black curve (GT), and the performance curve computed on 20 labels using the naive method from 5 random samples is shown in green (RND). Notice that the curves (in green) obtained from different samples vary a lot (although most predict perfect performance). **C**–**D**: same as A–B, but for the logres8 classifier on the DGT dataset (hand-picked as an example where SPE does not work well). **E**: Comparison of estimation error (area between curves) of SPE and naive method for 20 known labels and different datasets. The appearance of the markers denote the dataset (each dataset has multiple classifiers), and the lines indicate the standard error averaged over 10 trials. SPE almost always perform significantly better than the naive method.

where $\theta = \{\pi, \theta_0, \theta_1\}$, and $\pi \in [0, 1]$ is the mixture weight, i.e. $p(y_i = 1) = \pi$. The component densities p_0 and p_1 could be modeled parametrically by Normal distributions, Gamma distributions, or some other probability distributions appropriate for the given classifier (see Section 4.2 for a discussion about which class conditional distributions to choose). This approach of applying a generative model to score distributions, when all labels are known, has been used in the past to obtain error estimates on classifier performance [12, 9, 11], and for classifier calibration [1]. However, previous approaches require that the all items used to estimate the performance have been labeled.

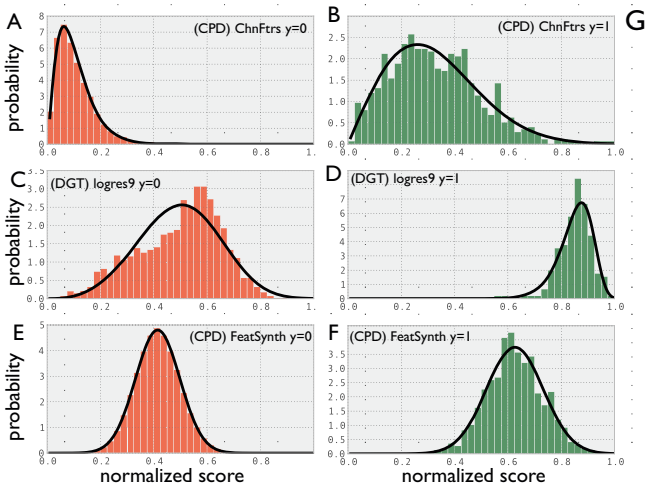
We suggest that it may be possible to estimate classifier performance even when only a fraction of the ground truth labels are known. In this case, the labels for the unlabeled items $i \in \mathcal{U}_t$ can be marginalized out,

$$p(S, Y_t \mid \theta) = \prod_{i \in \mathcal{U}_t} \left((1-\pi)p_0(s_i \mid \theta_0) + \pi p_1(s_i \mid \theta_1)\right) \times \prod_{i \in \mathcal{L}_t} \pi^{y_i}(1-\pi)^{1-y_i} p_{y_i}(s_i \mid \theta_{y_i}), \quad (2)$$

where $Y_t = \{y_i \mid i \in \mathcal{L}_t\}$. This allows the model to make use of the scores of unlabeled items in addition to the labeled items, which enables accurate performance estimates with only a handful of labels. Once we have the likelihood, we can take a Bayesian approach to estimate the parameters θ. Starting from a prior on the parameters, $p(\theta)$, we can obtain a posterior $p(\theta \mid S, Y_t)$ by using Bayes' rule,

$$p(\theta \mid S, Y_t) \propto p(S, Y_t \mid \theta)\, p(\theta). \quad (3)$$

Let us look at a real example. Figure 2a shows a histogram of the scores obtained from classifier on a public dataset (see Section 4 for more information about the datasets we use). At first glance, it is difficult to guess the performance of the classifier unless the oracle provides a lot of labels. However, if we *assume* that the scores follow a two-component mixture model as in (2), with a Gamma distribution for the $y_i = 0$ and a Normal distribution for the $y_i = 1$ component, then there is a only a narrow choice of θ that can explain the scores with high likelihood; the red and green curves in Figure 2a show such a high probability hypothesis. As we will see in the next section, the posterior on θ can be used to estimate the performance of the classifier.



dataset	1^{st}	2^{nd}	r.l.l.	3^{rd}	r.l.l.
(CPD) ChnFtrs [0]	g	ln	1.00	f-r	0.99
(CPD) ChnFtrs [1]	f-r	n	0.99	gu-r	0.97
(CPD) LatSvmV2 [0]	ln	g	1.00	f-r	0.99
(CPD) LatSvmV2 [1]	f-r	g	0.97	gu-r	0.96
(CPD) FeatSynth [0]	n	f-r	0.99	g	0.98
(CPD) FeatSynth [1]	n	g	1.00	f-r	0.98
(INR) LatSvmV2 [0]	ln	g	0.99	f-r	0.98
(INR) LatSvmV2 [1]	n	f-r	0.87	gu-l	0.73
(INR) ChnFtrs [0]	g	f-r	1.00	n	0.97
(INR) ChnFtrs [1]	f-r	n	0.97	g	0.84
(DGT) logres9 [0]	f-r	n	0.98	gu-l	0.96
(DGT) logres9 [1]	f-r	gu-l	0.99	n	0.99
(SAT) svm3 [0]	n	f-r	0.98	gu-r	0.84
(SAT) svm3 [1]	n	g	0.99	ln	0.98

Figure 4. Modeling class-conditional score densities by standard parametric distributions. **A**–**F**: Standard parametric distributions $p_y(s \mid \theta_y)$ (black solid curve) fitted to the class conditional scores for a few example datasets and classifiers. The score distributions are shown as histograms. In all cases, we normalized the scores to be in the interval $s_i \in (0, 1]$, and made the truncation at $s = 0$ for the truncated distributions. See Section 4.2 for more information. **G**: Comparison of standard parametric distributions best representing empirical class-conditional score distributions (for a subset of the 78 cases we tried). Each row shows the top-3 distributions, i.e. explaining the class-conditional scores with highest likelihood, for different combinations of datasets, classifiers and the class-labels (shown in brackets, $y = 0$ or $y = 1$). The distribution families we tried included (with abbreviations used in last three columns in parentheses) the truncated Normal (n), truncated Student's t (t), Gamma (g), log-normal (ln), left- and right-skewed Gumbel (g-l and g-r), Gompertz (gz), and Frechet right (f-r) distribution. The last and second to last column show the relative log likelihood (r.l.l.) with respect to the best (1^{st}) distribution. Two densities, truncated Normal and Gamma, are either top or indistinguishable from the top in all the datasets we tried.

3. Estimating performance

Most performance measures can be computed directly from the model parameters θ. For example, two often used performance measures are the precision $P(\tau; \theta)$ and recall $R(\tau; \theta)$ at a particular score threshold τ. We can define these quantities in terms of the conditional distributions $p_y(s_i \mid \theta_y)$. Recall is defined as the fraction of the positive, i.e. $y_i = 1$, examples that have scores above a given threshold,

$$R(\tau; \theta) = \int_{\tau}^{\infty} p_1(s \mid \theta_1)\, ds. \tag{4}$$

Precision is defined to be the fraction of all examples with scores above a given threshold that are positive,

$$P(\tau; \theta) = \frac{\pi R(\tau; \theta)}{\pi R(\tau; \theta) + (1 - \pi) \int_{\tau}^{\infty} p_0(s \mid \theta_0)\, ds}. \tag{5}$$

We can also compute the precision at a given level of recall by inverting $R(\tau; \theta)$, i.e. $P_r(r; \theta) = P(R^{-1}(r; \theta); \theta)$ for some recall r. Other performance measures, such as the equal error rate, true positive rate, true negative rate, sensitivity, specificity, and the ROC can be computed from θ in a similar manner.

The posterior on θ may be used to obtain confidence bounds on the performance of the classifier. For example, for some choice of parameters θ, the precision and recall can be computed for a range of score thresholds τ to obtain a curve (see solid curves in Figure 2b). Similarly, given the posterior on θ, the distribution of $P(\tau; \theta)$ and $R(\tau; \theta)$ can be computed for a fixed τ to obtain confidence intervals (shown as colored bands in Figure 2b). The same applies to the precision-recall curve: for some recall r, the distribution of precisions, found using $P_r(r; \theta)$ can be used to compute confidence intervals on the curve (see Figure 2c).

While the approach of estimating performance based purely on the estimate of θ works well in limit when the number of data items $N \to \infty$, it has some drawbacks when N is small (on the order of $10^3 - 10^4$) and π is unbalanced, in which case finite-sample effects come into play. This is especially the case when the number of positive examples is very small, say 10–100, in which case the performance curve will be very jagged. Since the previous approach views the scores (and the associated labels) as a finite sample from $p(S, Y \mid \theta)$, there will always be uncertainty in the performance estimate. When all items have been labeled by the oracle, the remaining uncertainty in the performance represents the variability in sampling (S, Y) from $p(S, Y \mid \theta)$. In practice, however, one question that is often asked is, "What is our best guess for the classifier performance on this particular test set?" In other words, we are interested in the sample performance rather than the population performance. Thus, when the oracle has labeled

the whole test set, there should not be any uncertainty in the performance; it can simply be computed directly from (S, Y).

To estimate the sample performance, we need to account for uncertainty in the unlabeled items, $i \in \mathcal{U}_t$. This uncertainty is captured by the distribution of the unobserved labels $Y'_t = \{y_i \mid i \in \mathcal{U}_t\}$, found by marginalizing out the model parameters,

$$\begin{aligned} p(Y'_t \mid S, Y_t) &= \int_\Theta p(Y'_t, \theta \mid S, Y_t)\, d\theta \\ &= \int_\Theta p(Y'_t \mid \theta) p(\theta \mid S, Y_t)\, d\theta. \end{aligned} \tag{6}$$

Here Θ is the space of all possible parameters. On the second line of (6) we rely on the assumption of a mixture model to factor the joint probability distribution on θ and Y'_t.

One way to think of this approach is as follows: imagine that we sample Y'_t from $p(Y'_t \mid S, Y_t)$. We can then use all the labels $Y = Y_t \cup Y'_t$ and the scores S to trace out a performance curve (e.g., a precision-recall curve). Every time we sample a different value for the labels Y'_t the performance curve will look slightly different. Thus, the posterior distribution on Y'_t in effect gives us a distribution of performance curves. We can use this distribution to compute quantities such as the expected performance curve, the variance in the curves, and confidence intervals. The main difference between the sample and population performance estimates will be at the tails of the score distribution, $p(S \mid \theta)$, where individual item labels can have a large impact on the performance curve.

3.1. Sampling from the posterior

In practice, we cannot compute $p(Y'_t \mid S, Y_t)$ in (6) analytically, so we must resort to approximate methods. For some choices of class conditional densities, $p_y(s \mid \theta_0)$, such as Normal distributions, it is possible to carry out the marginalization over θ in (6) analytically. In that case one could use collapsed Gibbs sampling to sample from the posterior on Y'_t, as is often done for models involving the Dirichlet process [14]. A more generally applicable method, which we will describe here, is to split the sampling into three steps: (a) sample $\bar{\theta}$ from $p(\theta \mid S, Y_t)$, (b) fix the mixture parameters to $\bar{\theta}$ and sample the labels Y'_t given their associated scores, and (c) compute the performance, such as precision and recall, for all score thresholds $\tau \in S$. By repeating these three steps, each of which is described in detail below, we can obtain a sample from the distribution over the performance curves.

The first step, sampling from the posterior $p(\theta \mid S, Y_t)$, can be carried out using importance sampling (IS). We experimented with Metropolis-Hastings and Hamiltonian Monte Carlo [15], but found that IS worked well for this problem, required less parameter tuning, and was much faster. In IS, we sample from a proposal distribution $q(\theta)$ in order to estimate properties of the desired distribution $p(\theta \mid S, Y_t)$. Suppose we draw M samples of θ from $q(\theta)$ to get $\bar{\Theta} = \{\bar{\theta}^1, \ldots, \bar{\theta}^M\}$. Then, we can approximate expected value of some function $g(\cdot)$ of θ using the weighted function evaluations, i.e. $E[g] \simeq \sum_{m=1}^{M} w_m g(\bar{\theta}^m)$. The weights $w_m \in W$ correct for the bias introduced by sampling from $q(\theta)$ and are defined as,

$$w_m = \frac{p(\bar{\theta}^m \mid S, Y_t)/q(\bar{\theta}^m)}{\sum_l p(\bar{\theta}^l \mid S, Y_t)/q(\bar{\theta}^l)}. \tag{7}$$

For the datasets in this paper, we found that the state-space around the MAP estimate[1] of θ,

$$\theta^\star = \arg\max_\theta p(\theta \mid S, Y_l), \tag{8}$$

was well approximated by a multivariate Normal distribution. Hence, for the proposal distribution we used,

$$q(\theta) = \mathcal{N}(\theta \mid \mu_q, \Sigma_q). \tag{9}$$

To simplify things further, we used a diagonal covariance matrix, Σ_q. The elements along the diagonal of Σ_q were found by fitting a univariate Normal locally to $p(\theta \mid S, Y_t)$ along each dimension of θ while the other elements were fixed at their MAP-estimates. The mean of the proposal distribution, μ_q, was set to the MAP estimate of θ.

We now have all steps needed to estimate the performance of the classifier, given the scores S and some labels Y_t obtained from the oracle:

1. Find the MAP estimate μ_q of θ using (8).
2. Fit a proposal distribution $q(\theta)$ to $p(\theta \mid S, Y_t)$ locally around μ_q.
3. Sample M instances of θ, $\bar{\Theta} = \{\bar{\theta}^1, \ldots, \bar{\theta}^M\}$, from $q(\theta)$ and calculate the weights $w_m \in W$.
4. For each $\bar{\theta}^m \in \bar{\Theta}$, sample the labels for $i \in \mathcal{U}_t$ to get $\bar{Y}'_t = \{\bar{Y}'_{t,1}, \ldots, \bar{Y}'_{t,M}\}$.
5. Estimate performance measures using the scores S, labels $\bar{Y}_{t,m} = Y_t \cup \bar{Y}'_{t,m}$ and weights $w_m \in W$.

4. Experiments

4.1. Datasets

We surveyed the literature for published classifier scores with ground truth labels. One such dataset that we found was the Caltech Pedestrian Dataset[2] (CPD), for which both

[1] We used BFGS-B [4] to carry out the optimization. To minimize the issue of local maxima, we used multiple starting points.

[2] Downloaded from http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/.

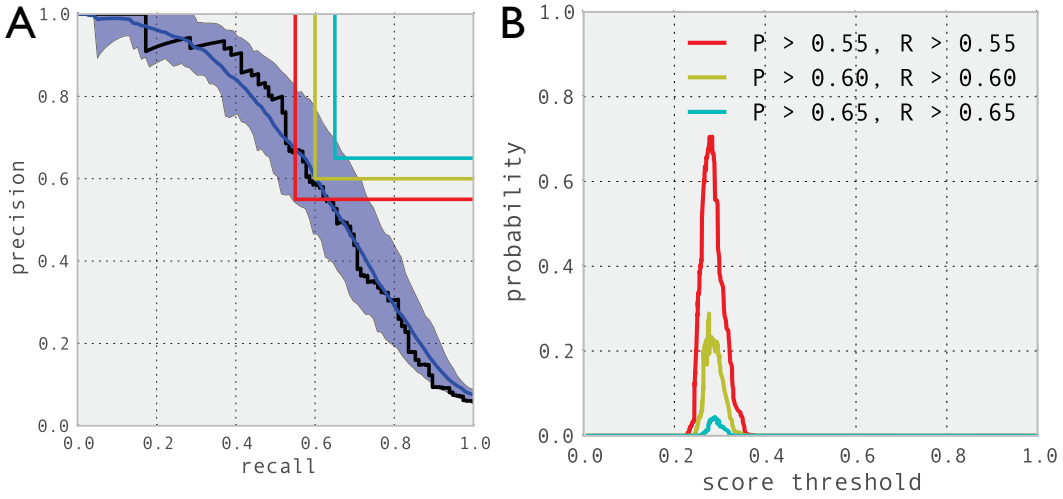


Figure 5. Recalibrating the classifier by estimating the probability that a condition is met. **A**: The conditions in panel B shown as colored "boxes," e.g. the yellow curve shows the condition that the precision $P > 0.5$ and recall $R > 0.5$. The blue curve and confidence band show SPE applied to the ChnFtrs detector on the CPD dataset with 100 observed labels (black curve is ground truth). **B**: Probability that the conditions shown in A are satisfied for different score thresholds. Based on a curve like this, a practitioner can "recalibrate" a pre-trained classifier by picking a threshold for new dataset such that some pre-defined criteria (e.g. in terms of precision and recall) are met.

detector scores and ground truth labels are available for a wide variety of detectors [7]. Moreover, the CPD website also has scores and labels available, using the same detectors, for other pedestrian detection datasets, such as the INRIA (abbr. INR) dataset [5].

We made use of the detections in the CPD and INR datasets as if they were classifier outputs. To some extent, these detectors are in fact classifiers, in that they use the sliding-window technique for object detection. Here, windows are extracted at different locations and scales in the image, and each window is classified using a pedestrian classifier (with the caveat that there is often some extra post-processing steps carried out, such as non-maximum suppression to reduce the number of false positive detections). For our experiments, we show the results on detectors and datasets to highlight both the advantages and drawbacks with using SPE. To make experiments go faster, we sampled the datasets randomly to have between 800–2,000 items. See [7] for references to all detectors.

To complement the pedestrian datasets, we also used a basic linear SVM classifier and a logistic regression classifier on the "optdigits" (abbr. DGT) and "sat" (SAT) datasets from the UCI Machine Learning Repository [10]. Since both datasets are multiclass, but our method only handles binary classification, we chose one category for $y = 1$ and grouped the others into $y = 0$. Thus, each multi-class dataset was turned into multiple binary datasets. Planned future work includes extending our approach to multiclass classifiers. In the figures, the naming convention is as follows: "svm3" is used to mean that the SVM classifier was used with category 3 in the dataset being assigned to the $y = 1$ class, and "logres9" denotes that the logistic regression classifier was used with category 9 being the $y = 1$ class, and so on. The datasets had 1,800–2,000 items each.

4.2. Choosing class conditionals

Which distribution families should one use for the class conditional $p_y(s \mid \theta_y)$ distributions? To explore this question, we took the classifier scores and split them into two groups, one for $y_i = 0$ and one for $y_i = 1$. We used MLE to fit different families of probability distributions (see Figure 4 for a list of distributions) on 80% of the data (sampled randomly) in each group. We then ranked the distributions by the log likelihood of the remaining 20% of the data (given the MLE-fitted parameters). In total, we carried out this procedure on 78 class conditionals from the different datasets and classifiers.

Figure 4G shows the top-3 distributions that explained the class-conditional scores with highest likelihood for a selection of the datasets and classifiers. We found that the truncated Normal distribution was in the top-3 list for 48/78 dataset class-conditionals, and that the Gamma distribution was in the top-3 list 53/78 times; at least one of the two distributions were always in the top-3 list. Figure 4A–F show some examples of the fitted distributions. In some cases, like Figure 4C, a mixture model would have provided a better fit than the simple distributions we tried. That said, we found that truncated Normal and Gamma distributions were good choices for most of the datasets.

Since we use a Bayesian approach in equation (3), we

must also define a prior on θ. The prior will vary depending on which distribution is chosen, and it should be chosen based on what we know about the data and the classifier. As an example, for the truncated Normal distribution, we use a Normal and a Gamma distribution as priors on the mean and standard deviation respectively (since we use sampling for inference, we are not limited to conjugate priors). As a prior on the mixture weight π, we use a Beta distribution.

In some situations when little is known about the classifier, it makes sense to try different kinds of class-conditional distributions. One heuristic, which we found worked well in our experiments, is to try different combinations of distributions for p_0 and p_1, and then choose the combination achieving the highest maximum likelihood on the labeled and unlabeled data.

4.3. Applying SPE

Figure 3 shows SPE applied to different datasets. The left-most plots show the estimation error, as measured by the area between the true and predicted precision-recall curves, versus the number of labels sampled. The datasets in Figure 3A–B and C–D were chosen to highlight the strengths and weaknesses of using SPE. Figure 3A shows SPE applied to the ChnFtrs detector in the CPD dataset. Already at 20 sampled labels, the estimate is very close (see Figure 3B). In a few cases, e.g. in Figure 3C–D (logres8 on the DGT dataset), SPE does not fare as well. While SPE performs as well as the naive method in terms of estimation error, the score distribution is not well explained by the assumptions of the model, so there is a bias in the prediction. That said, despite the fact that SPE is biased in Figure 3D, it is still far better than the naive method for 100 labels. Ultimately, the accuracy of SPE depends on how well the score data fit the assumptions in Section 2.

Figure 3E compares the estimation error of SPE to the naive method for different datasets, when only 20 labels are known. In almost all cases, SPE performs significantly better. Moreover, the variances in the SPE estimates are smaller than those of the naive method.

4.4. Classifier recalibration

Applying SPE to a test dataset allows us to "recalibrate" the classifier to that dataset. Unlike previous work on classifier calibration [1, 17], SPE does not require all items to be labeled. For each unlabeled data item, we can compute the probability that it belongs to the $y = 1$ class by calculating the empirical expectation from the samples, i.e. $\hat{p}(y_i = 1) = E[y_i = 1 \mid S, Y_t]$.

Similarly, we can choose a threshold τ to use with the classifier $\bar{h}(x_i; \tau)$ based on some pre-determined criteria. For example, the requirement might be that the classifier performs with recall $R(\tau) > \hat{r}$ and precision $P(\tau) > \hat{p}$. In that case, we define a condition $C(\tau) = [R(\tau) > \hat{r} \wedge P(\tau) > \hat{p}]$. Then, for each τ, we find the probability that the condition is satisfied by calculating the expectation $\hat{p}(C(\tau) = 1) = E[C(\tau)]$ over the unlabeled items Y'_t. Figure 5 shows the probability that $C(\tau)$ is satisfied at different values of τ. Thus, this approach can be used to choose new thresholds for different datasets.

5. Related work

Previous approaches for estimating classifier performance with few labels falls into two categories: stratified sampling and active estimation using importance sampling. Bennett and Carvalho [2] suggested that the accuracy of classifiers can be estimated cost-effectively by dividing the data into disjoint strata based on the item scores, and proposed an online algorithm for sampling from the strata. This work has since been generalized to other classifier performance metrics, such as precision and recall [8]. Sawade et al. proposed instead to use importance sampling to focus labeling effort on data items with high classifier uncertainty, and applied it to standard loss functions [19] and F-measures [18]. Both of these approaches assume that the classifier threshold τ is fixed (see Section 2) and that a single scalar performance measure is desired. SPE does not fix τ, and can thus be used to show the *tradeoff* between different performance measures in the form of performance curves.

Fitting mixture models to the class-conditional score distributions has been studied in previous work with the goal of obtaining smooth performance curves. Gu et al. [11] and Hellmich et al. [12] showed how a two-component Gaussian mixture model can be used to obtain accurate ROC curves in different settings. Erkanli et al. [9] extended this work by fitting mixtures of Dirichlet process priors to the class-conditional distributions. This allowed them to provide smooth performance estimates even when the class-conditional distributions could not be explained by standard parametric distributions. Similarly, previous work on classifier calibration has involved fitting mixture models to score distributions [1, 17]. In contrast to previous work, which require all data items to be labeled, SPE also makes use of the unlabeled data. This semisupervised approach allows SPE to estimate classifier performance with very few labels, or when the proportions of positive and negative examples are very unbalanced.

6. Discussion

We explored the problem of estimating classifier performance from few labeled items. We propose a method, Semisupervised Performance Evaluation (SPE), based on modeling the scores of classifiers with mixtures of densities. SPE estimates performance curves even when a very small number (none in the limit) of the samples are labeled. Fur-

thermore, it produces bounds on the performance curves. The bounds shrink as we add more level examples, allowing a user to trade off labeling effort and uncertainty. A sampling scheme based on importance sampling enables efficient inference.

One disadvantage with using an approach like SPE is that there are no guarantees that the assumption of standard parametric distributions will hold for any dataset, something we hope to address in future work. Furthermore, strong model assumptions always bear the risk of underestimating errors. However, using four public datasets, and multiple classifiers, we showed that classifier score distributions are often well approximated by standard two-component mixture models in practice.

This line of research opens up many interesting avenues for future exploration. For example, is it possible to do unbiased active querying, so that the oracle is asked to label the most informative examples? One possibility in this direction would be to employ importance weighted active sampling techniques [3, 6], so similar in spirit to [19, 18] but for performance curves. Another direction for investigation is extending SPE to multi-component mixture models and multiclass problems. Multi-component mixture models, for example, would overcome the assumption of unimodal score conditionals. That said, as shown by our experiments, SPE already works well for a broad range of classifiers and datasets, and can estimate classifier performance with as few as 10 labels (see Figure 2).

7. Acknowledgements

This work was supported by National Science Foundation grants 0914783 and 1216045, NASA Stennis grant NAS7.03001, ONR MURI grant N00014-10-1-0933, and gifts from Qualcomm and Google.

References

[1] P. N. Bennett. Using asymmetric distributions to improve classifier probabilities: A comparison of new and standard parametric methods. Technical report, Carnegie Mellon University, 2002. 3, 7

[2] P. N. Bennett and V. R. Carvalho. Online stratified sampling: evalutating classifiers at web-scale. In *CIKM*, 2010. 7

[3] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *ICML*, 2009. 8

[4] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995. 5

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *ICCV*, 2005. 1, 2, 6

[6] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *ICML*, 2008. 8

[7] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 99, 2011. 1, 2, 6

[8] G. Druck and A. McCallum. Toward interactive training and evaluation. In *CIKM*, 2011. 7

[9] A. Erkanli, M. Sung, E. J. Costello, and A. Angold. Bayesian semi-parametric ROC analysis. *Statist. Med.*, 25:3905–3928, 2006. 3, 7

[10] A. Frank and A. Asuncion. UCI machine learning repository, 2010. 6

[11] J. Gu, S. Ghosal, and A. Roy. Bayesian bootstrap estimation of ROC curve. *Statist. Med.*, 27:5407–5420, 2008. 3, 7

[12] M. Hellmich, K. R. Abrams, and A. J. Sutton. Bayesian Approaches to Meta-analysis of ROC Curves. *Med. Decis. Making*, 19:252–264, 1999. 3, 7

[13] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012. 1

[14] S. N. MacEachern. Estimating normal means with a conjugate style dirichlet process prior. *Communications in Statistics B*, 23(3):727–741, 1994. 5

[15] R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. L. Jones, , and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman & Hall / CRC Press, 2010. 5

[16] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3):103–134, 2000. 2

[17] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74. MIT Press, 1999. 7

[18] C. Sawade, N. Landwehr, S. Bickel, and T. Scheffer. Active risk estimation. In *ICML*, 2010. 7, 8

[19] C. Sawade, N. Landwehr, and T. Scheffer. Active Estimation of F-Measures. In *NIPS*, 2010. 7, 8

[20] M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001. 2

[21] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin–Madison, 2005. 2