Improving Prediction Accuracy Using a Calibration Postprocessor

David B. Rosen **Harry B. Burke**
Department of Medicine
New York Medical College
Valhalla, New York 10595 USA
<d.rosen@ieee.org> <burke@unr.edu>

Philip H. Goodman
University of Nevada School of Medicine
77 Pringle Way #H1-166
Reno, Nevada 89520 USA
<goodman@unr.edu>

ABSTRACT. We propose using a *calibration* module to augment the architecture of a classifier, mapping numerically-arbitrary ordinal discriminant values to accurate probability predictions. Results from the application of a simple memory-based nonparametric classifier to a large breast cancer data set suggest that the bias resulting from a very large smoothing (regularization) parameter is sometimes manifested largely as miscalibration, rather than impairing discrimination. So even when the original classifier is itself designed to estimate probabilities, the separate calibration step allows that classifier to focus on improving its ability to *discriminate* between two classes, by increasing its smoothing or regularization, attaining lower variance, and thus improving overall performance. The calibration method is motivated here by setting thresholds using decision theory.

1 PROBABILITY ESTIMATION VS. DISCRIMINATION

When methods providing a single real-valued prediction [1] are used for two-class[2] discrete classification, one often applies some threshold as a "postprocessing" step. If the continuous prediction is intended to be of an input-conditional class probability[3], then one can specify the threshold in advance, given some assumption about the relative "cost" of the two types of misclassification error. The most common (though not always most appropriate) assumption is of equal costs, leading to a threshold of $\frac{1}{2}$. A prediction that is a probability estimate has the advantage of being meaningful even before thresholding, so it can easily be used by (perhaps several) decision-makers regardless of the costs entering into their own (perhaps many) decision problems. For example, if a weather forecast states that "it will rain today with probability 0.2", various decisions can be made, such as whether to carry an umbrella or cancel an outdoor event, any of which might have different costs entering into them and thus use different decision thresholds.

Alternatively, the continuous prediction of a classifier may be merely a *discriminant*, in which case the optimal threshold may have to be chosen empirically for a particular decision problem. This threshold selection process is a type of *calibration*, as it implicitly labels a particular point on the previously numerically-arbitrary (i.e. ordinal) discriminant scale as a specific probability (e.g. $\frac{1}{2}$ for an equal-costs decision).

2 MEMORY-BASED GAUSSIAN KERNEL METHOD

In this paper we examine the choice of the kernel width (smoothing or regularization parameter) in a local memory-based nonparametric classifier. We use gaussian kernels $N(\mathbf{x};\, \mu, \sigma^2)$ in class-conditional kernel (a.k.a. Parzen window) density estimation (Rosenblatt, 1956), with a single (common to both classes) scalar kernel width σ. The posterior probability estimate of interest is then $\hat{p}(\text{class } 1|\mathbf{x}) = \left[\sum_{k \text{ in class } 1} N(\mathbf{x};\, \mathbf{X}_k, \sigma^2)\right] / \left[\sum_{\text{all } k} N(\mathbf{x};\, \mathbf{X}_k, \sigma^2)\right]$, when presented with input feature vector $\mathbf{x}$, where $\mathbf{X}_k$ is the input feature vector of the kth stored "training" case. The formula above is a special case of the Nadaraya(1964)-Watson(1964) kernel-weighted average of training-set target values, or in a more parametric context, normalized radial basis functions (Moody & Darken, 1989). Hardware-oriented neural network implementations exist (Specht, 1991). At the outset, we normalize each input component (feature) to have a standard deviation of 1 in the training data.

[1] i.e. response to an input feature pattern during recall (performance)

[2] In the interest of simplicity, we limit our discussion to two-class problems in this paper.

[3] Called a posterior probability in the pattern recognition literature, if the classifier used Bayes Theorem to calculate it from a prior probability and the class-conditional probabilities of the input.

3 PREDICTIONS AS PROBABILITY ESTIMATES

The accuracy of a probability estimator can be evaluated on a data set using a *strictly proper scoring rule*, defined as a loss function whose expectation is optimized by — and only by — the true probability that generated the data (or at least generated your expectation) (Savage, 1971). Familiar examples include the quadratic (squared error or Brier score) and logarithmic (e.g. cross-entropy) loss functions. Any particular decision problem loss, such as the misclassification error (zero or one for each prediction), is a proper scoring rule, i.e. optimized in expectation by the true probability, but not *strictly* proper, so its expecation may be optimized by other probability estimates in addition to the correct one. In particular, such a loss function is only sensitive to whether the estimate is above or below a single threshold. For this reason among others, the smoothing parameter is often chosen (Hand, 1982; Silverman, 1986; Härdle, 1991) to minimize the squared error.

To prevent overfitting the training set, cross-validation can be used for a reasonably unbiased estimate of the true (generalization) squared error. We used leave-out-one cross-validation, which entails using all the training data *except* a given case, in order to calculate the prediction for that case, and calculating the squared error of these *jacknife* predictions. Because for each case of a given class, the effective training set (of size $n - 1$) has one fewer case of that class, the prediction is slightly biased towards the other (i.e. wrong) class. We correct for this by arranging for the total weight given to the cases of each class in the kernel-weighted average to always reflect the overall prevalence of that class. Ordinarily this correction would be considered negligible; its importance to us will be explained in Section 4.

The kernel width controls the degree of regularization (shrinkage). A narrow kernel leads to high variance but low bias (undersmoothed, overfit); a wide kernel does the opposite. The (in)accuracy as a function of regularization is ordinarily expected to be U-shaped.

We report experiments conducted using subsets of the Commission on Cancer Patient Care Evaluation (PCE) breast cancer data. The PCE data were collected by the American College of Surgeons (ACOS), jointly sponsored with the American Cancer Society, by requesting data from individual tumor registries on the first 25 cases of first diagnosis breast cancer (among others) seen in 1983 at each ACOS-accredited hospital in the United States. In particular, we used 5169 training cases and 3102 test cases, chosen for their lack of missing variable values. For each case the data describe various clinical and pathologic prognostic factors (none from images or signals) as determined at the time of diagnosis, coded as 54 features. We used 5-year survival as our binary target class variable. A U-shaped average(-over-cases) squared error does appear in Figure 1, with an "optimal" width of about $\sigma = 2$ input standard deviations. However, the squared error there is still poor compared, for example, to a vanilla multi-layer perceptron (horizontal line indicated). Vanilla logistic regression does almost as well as the MLP; we have not thus far seen strong evidence of nonlinearity in this data set.

4 PREDICTIONS AS DISCRIMINANT SCORES

The squared error (or another proper scoring rule) measures the inaccuracy in the predictions as probability estimates. To treat the predictions as mere discriminants, we need an accuracy criterion that depends only on the *order* of the predictions (which predictions are greater than which others), but not on their numerical values per se.

One such criterion is the c index. We look at (all possible) pairs of training cases such that one is in class 1 (survived) and the other in class 0 (did not survive). The better the predictions are as a discriminant scale, the more often that the discriminant (assumed originally to be the estimated probability of class 1) is greater for the class 1 case than for the class 0 case. The c index is defined as the fraction of pairs for which this is true. It also equals the area under the empirical receiver operating characteristic (ROC) curve[4], which is widely used in signal detection and medical decision analysis. If the predictions were unrelated to the true class, c would be equal to $\frac{1}{2}$; if all the predictions for class 1 cases were greater than all those for class 0 cases, we would have perfect discrimination (separation) of the classes and $c = 1$.

Figure 2 shows $1 - c$ (so lower is better, as with squared error) as a function of kernel width σ. Note that it appears to improve in a monotone fashion, without the bias-variance U shape. For example, width 32 has a significantly ($p < .001$) *better* $1 - c$ than width 2. In contrast, the squared error showed width 32 being significantly ($p = .01$) *worse* than width 2; the latter was the best width by this criterion.

As the kernel width increases, all the training points are given nearly the maximum weight (they are well within the width of the kernel), so all predictions approach the overall prevalence of class 1 in the training set (about 0.8). Figure 3 is a scatterplot where each dot represents one case, showing the width$=$ 2 prediction vs. the width$=$ 32 prediction

[4]plot $\Pr(\text{decision} = 1|\text{class} = 1)$ vs. $\Pr(\text{decision} = 1|\text{class} = 0)$ for all decision thresholds

for that case. Although a large kernel width compresses the predictions into a narrow range, Figure 4 zooms in on the horizontal axis to show that the predictions are strongly correlated at the different widths, demonstrating how the relative order can be preserved even when the numerical values are changed a great deal.

We can now see the importance of the prevalence correction described in Section 3. At large kernel widths, small differences in the prediction carry a lot of information; these differences would often be completely washed out by the effect of the prevalence differences in the $n-1$ remaining training cases according to the class of the case being left out.

5 CALIBRATING A SINGLE DECISION THRESHOLD

We use a hypothetical decision problem to illustrate how treating the predictions as a discriminant rather than taking them literally as a probability can give better results. Suppose that for current breast cancer patients it must be decided whether a new treatment Z will be given. We have some external knowledge about Z: it saves about half the patients who otherwise would have died. We assign a "cost" of 20 to death, and a "cost" of 1 to the side effects and complication risks of treatment Z itself.

No patients in our collected data received Z (since it is new), but presumably, better predictions of patient outcome without Z will enable us to make better decisions about when it is worth the risk to give Z. We wish to use the existing

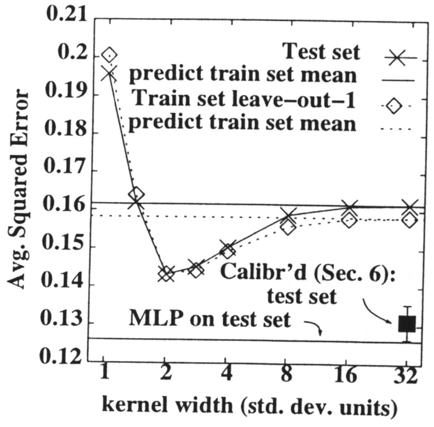

Figure 1: An (in)accuracy measure for probabilities.

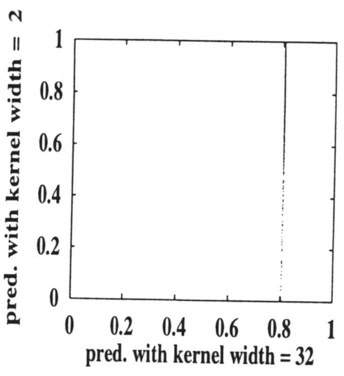

Figure 3: Jacknife predictions on each training case.

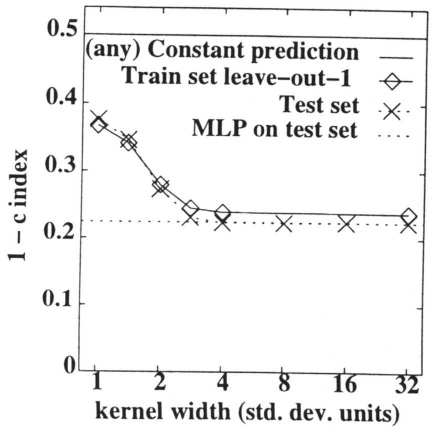

Figure 2: An (in)accuracy measure for discrimination only.

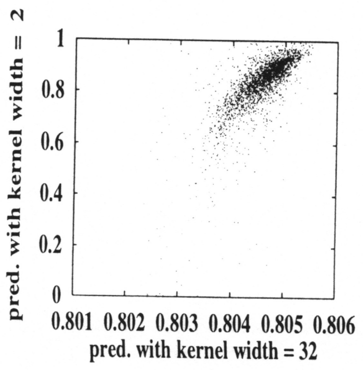

Figure 4: Expand the important region of Figure 3

Table 1a

give Z?	width= 2 (thresh= .9) surv w/o Z	width= 2 (thresh= .9) die w/o Z	width= 32 (thresh = .9) surv w/o Z	width= 32 (thresh = .9) die w/o Z
don't:	777	75	0	0
do:	1696	554	2473	629
	regret = 2371		**regret = 2473**	

Table 1b

width= 2 thresh= .9028 surv w/o Z	width= 2 thresh= .9028 die w/o Z	width= 32 thresh = .804727 surv w/o Z	width= 32 thresh = .804727 die w/o Z
707	70	1210	100
1766	559	1263	529
regret = 2396		**regret = 2163**	

test set to determine how good (in expectation) our decisions would have been on those patients if Z had been available at that time.

If a given patient in our data set did in fact die, then they "would die without treatment Z". If she *had* received Z, the outcome would have been survival with probability 0.5 . So the *expected* loss for this patient due to death is half that of certain death. Thus, we calculate the total loss matrix as:

	would survive without trtmt. Z	would die without trtmt. Z
don't give trtmt. Z	0	20
do give trtmt. Z	1	$20 \times 0.5 + 1 = 11$

.

Decision theory then gives the optimal decision rule: *Give treatment Z if and only if the probability of surviving without it is less than* 0.9, i.e. decision threshold is 0.9[5] In our test set there is an inevitable loss of 629 deaths $\times 11 = 6919$. Define *regret* as loss minus inevitable loss, so that a regret of zero is obtained for perfect prediction.

If we treat the predictions as probability estimates, we will naively apply threshold 0.9 to the two classifiers (having kernel widths 2 and 32 respectively). The resulting two-by-two "misclassification" (actually misdecision) matrices (showing numbers of cases) for the test set, and resulting total regret, are shown in Table 1a, indicating that we should prefer the predictions of the width= 2 classifier — as with the earlier squared error result.

Now consider treating the predictions as (merely) a discriminant. We optimize the threshold to give the least least regret on the *training* set (jacknife). Table 1b shows the chosen threshold value, misdecision matrix on the independent test set, and resulting total regret. With "recalibrated" thresholds, we favor width= 32 over width= 2. With a regret of 2163, we have also improved from the best non-recalibrated regret of 2371, demonstrating the benefit of optimizing the threshold. It can be shown that we would have chosen the same discriminant threshold for any decision loss matrix that is "equivalent" in the sense that it has the same probability threshold.

6 CALIBRATION OF ALL PREDICTION VALUES

In order to regain the advantage of obtaining a probability estimate rather than a particular thresholded decision, we need to recalibrate the entire prediction scale, i.e. find a continuous (and non-decreasing) mapping from the original prediction to a probability prediction.

The previous Section indicates that the calibration should map a discriminant value of about .804727 to a probability estimate of .9. If we were to sweep through decision problem probability thresholds continuously from 0 to 1, optimizing the discriminant threshold as we went, we see the regret change only when the discriminant threshold crosses over the discriminant value for a case, changing the decision for that case. So it suffices to construct an ordered lookup table, with one entry per training case, giving the discriminant and corresponding probability estimate. Between these values we would have to interpolate.

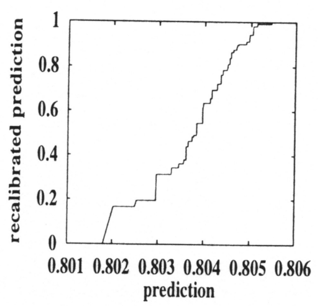

Figure 5: Calibration curve for width=32 (from train set jacknife predictions).

[5]Note that a symmetric-loss decision (misclassification rate) problem would render these predictions almost useless given how few predictions would be less than the threshold of $\frac{1}{2}$.

There are at least two strategies for efficiently generating the calibration lookup table with one entry for each calibration data set point, without explicitly performing any nonlinear optimizations. One is to start with an identity calibration where each original prediction is mapped to itself (or in fact to the corresponding element of any other nondecreasing sequence), and go through the cases, moving values towards the outcome values to the maximum extent allowed while maintaining the nondecreasing ordering (Rosen, 1994). The second strategy is to start with the best possible calibration values *ignoring* the monotonicity constraint, namely the true outcomes (target values) in this data set, and then go through the cases sequentially, enforcing monotonicity. This is the strategy employed by the "Pool Adjacent Violators" (Barlow et al., 1972) algorithms used to perform monotone regression; the Appendix describes how we implemented one such algorithm.

Figure 5 shows the resulting calibration calculated solely from our training set (jacknife predictions). We incorporated this mapping[6] into our prediction system as a postprocessor. The resulting predictions on the still-independent *test* set have an average squared error of $= .131$ [$\pm .0046$] (point with standard-error bar in Figure 1), which is no longer significantly different from the MLP result.

7 DISCUSSION

In the present experiments, much of the "bias" due to oversmoothing (over-regularization) could be calibrated away by a postprocessing module. Greater smoothing reduced variance, and so overall accuracy improved.

We view the presence of a calibration module within the architecture of a classification system as more important than the specific calibration algorithms it may employ. The calibration algorithm we used is nonparametric and memory-based, and contains no architectural or regularization parameters. Alternatively, it should be straightforward to fit a reasonably-small parametric network to the points of the calibration curve. However, one would would to be careful to enforce monotonicity and to prevent the possibility (as for example with a linear calibration) of calibrating a future prediction to a "probability" outside $[0, 1]$.

As for the choice of the classification algorithm itself, a simple local kernel classifier was well suited to these first experiments because it is very easy to understand and interpret. Further work will be needed to establish whether this effect is important across classifier algorithms/architectures, and across characteristics of the application domain, such as their nonlinearity.

Despite the pedagogic focus on decision thresholds in this treatment, the same calibration algorithms can be applied to continuous-target regression problems as well, considering the original prediction to be related to the prediction of interest through a nondecreasing calibration curve.

APPENDIX: Implementing An Algorithm to Generate the Calibration Table

This is essentially an implementation of an "Up-and-Down Blocks" algorithm, which has been attributed to J. B. Kruskal and is essentially a type of "Pool Adjacent Violators" algorithm (Barlow et al., 1972).

The array `orig[]` are the *sorted* original predictions for the calibration data, which in our case is the training set but with jacknife predictions, for the n training cases. The lookup table will comprise n pairs (`orig`[0],`cal`[0]), (`orig`[1],`cal`[1]), etc. This table (with simple interpolation) will then become the fixed calibration postprocessor for future recall performance by the system. Values `cal`[0 ... n - 1] are initialized to corresponding values `target`[0 ... n - 1], which are each 0 or 1 to indicate the true class of the case.

Now we must modify `cal[]` just enough (away from the target values) to enforce the stringent constraint: calibration must be a nondecreasing (and single-valued) function. When the target value *decreases* (from 1 to 0) from case i to case $i + 1$, the best we can do under the constraint is assign *equal* calibrated predictions to these two cases. In this way, training cases agglomerate with their neighbors into sets of consecutive case indices. The `cal[]` values for all members of a given such set are constrained to be equal, and the best value to give them is the mean of the corresponding set of target values, which optimizes any proper scoring rule on this set. There are no arbitrary groupings specified in advance — the algorithm groups cases only where necessary in order to solve the constrained optimization problem.

We employ an ordered list of index sets partitioning indices $\{0 \ldots n-1\}$ into length(list) mutually exclusive and exhaustive sets. The list is initialized to contain n index sets, each such set containing a single distinct index: $(\{0\}, \{1\}, \ldots, \{n-1\})$. Then to enforce single-valuedness, any adjacent sets in the list having exactly the same `orig[]` value are *merged*, meaning that two sets are replaced by a single one containing the union of indices from both (so the list of sets gets shorter by one set), and all `cal`[the merged set] values are set to the mean among these values.

The list of course has a `first_set` and `last_set`. There is a pointer to one set in the list, this set being termed the

[6]with the crude constant (not even linear) interpolation: truncation to the next lower table entry

Procedure 1

```
> while ( current_set is not last_set ) do
> > if cal[ member of current_set ] >= cal[ member of next_set ] then
> > > merge next_set with current_set
> > > current_set pointer points to that merged set
> > > decrement current_set pointer (if current_set is not first_set)
> > else increment current_set pointer
```

current_set. That pointer is initialized so that current_set is first_set. Procedure 1 is then executed. Recall that cal[member of a set] does not depend on *which* member of the set, since we ensure that all these components of cal[] remain equal. The decrement step serves to allow the merged current_set (with its new cal[] values) to be merged with the previous set if necesary, and so on back down the list. Note also that the mean of cal[current_set] values is always the same as the mean of target[current_set] values.

Acknowledgements

Supported in part by the U.S. Army Medical Research Command (DAMD-17-94-J4383), U.S. Agency for Health Care Policy and Research (AHCPR-HS06830), American Cancer Society, and the American College of Surgeons. Some of this research was begun at Boston University, supported in part by U.S. DARPA (AFOSR 90-0083), and more was begun by all the authors while at the University of Nevada School of Medicine, in the Department of Medicine. That department, and the office of the Dean of the Graduate School, provided additional funding.

References

Barlow, R. E., Bartholomew, D. J., Bremner, J. M., & Brunk, H. D. (1972). *Statistical Inference under Order Restrictions.* Wiley, London.

Hand, D. J. (1982). *Kernel Discriminant Analysis.* Research Studies Press, Chichester.

Härdle, W. (1991). *Smoothing Techniques With Implementation in S.* Springer-Verlag, New York.

Moody, J., & Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1**, 281–294.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability and its Appl.*, **10**, 186–190.

Rosen, D. B. (1994). Ordinal discrimination index for any proper scoring rule. *Medical Decision Making*, **14**, 440. Published Abstract.

Rosenblatt, M. (1956). Remarks on some non-parametric estimates of a density function. *Ann. Math. Stat.*, **27**, 642–669.

Savage, L. J. (1971). Elicitation of personal probabilities and expectations. *J. American Stat. Assoc.*, **66**, 783–801.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London.

Specht, D. F. (1991). A general regression neural network. *IEEE Tr. Neural Networks*, **2**, 568–576.

Watson, G. S. (1964). Smooth regression analysis. *Sankhya*, **A26**, 359–372.