Calibration for probabilistic classification

Nick Normandin

Overview

The problem

some classifiers produce well-calibrated probabilities

- discriminant analysis
- logistic regression

others don't

- naive bayes
- SVMs
- anything with boosting
- tree methods
- sometimes neural networks

First of all, who cares?

1. people with asymmetric misclassification costs
2. people who are going to use the scores in post-processing
3. people who want to compare model outputs on a fair basis

Definitions: "classification"

in general, a classifier is a mapping function f such that

$$f : \vec{x} \mapsto c$$

where $\vec{x} \in \mathbb{R}^P$, but we're mostly interested in the intermediate step in where the function produces some membership score s_i for each instance $\vec{x}_i$

Definitions: "well-calibrated"

- for a model f and score s_i to be well-calibrated for class c_i, the empirical probability of a correct classification $P(c_i|f(c_i|x_i) = s_i)$ must converge to $f(c_i|x_i) = s_i$
- **example**: when $s_i = 0.9$, the probability of a correct classification should converge to $P(c_i|s_i = 0.9) = 0.9$. Otherwise, this isn't *really* a 'probability.'

Definitions: "calibration"

the calibration process is a separate mapping such that

$$g : s_i \mapsto P(c_i|s_i)$$

it's really important to note that we're fitting another model on top of our model output, where your feature matrix is just the vector of probability scores $\vec{s}$ and the target variable is the vector of true class labels $\vec{y} \in \{0, 1\}$

Visualizing calibration

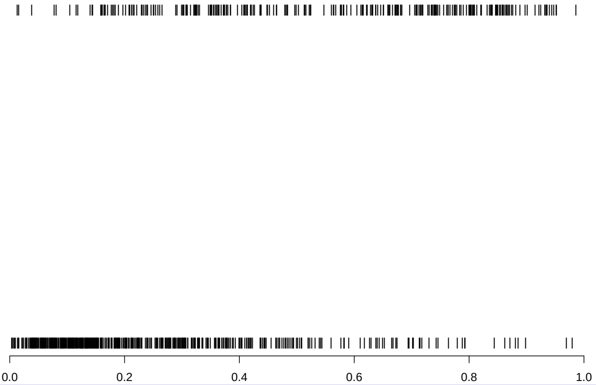
Visualizing calibration

```
# train SVM w/ linear kernel on Pima Indian Diabetes data
m <- train(x = PimaIndiansDiabetes[,1:8],
           y = PimaIndiansDiabetes[,9], tuneLength = 1,
           method = 'svmLinear',
           trControl = trainControl(method = 'cv',
                                    savePredictions = T,
                                    classProbs = T))

pred <- m$pred[order(m$pred$rowIndex),]
result <- data.table(prob = pred$pos,
                     class = ifelse(pred$obs == 'pos', 1, 0))
```

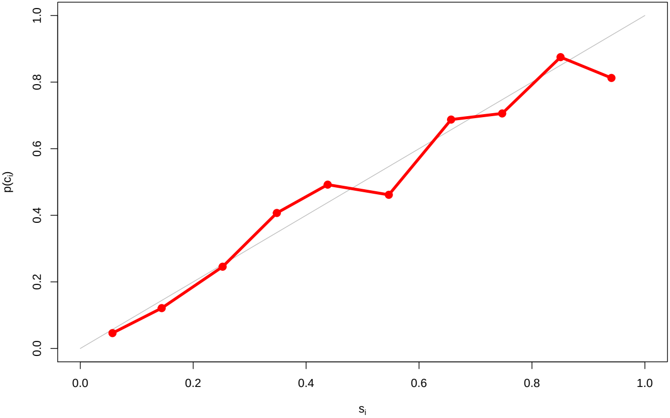
Visualizing calibration

plotting the model class score s_i vs the true label y_i. Is this a useful representation?



Reliability plots

(1) Bin predictions by s_i (x-axis), **(2)** calculate $p(c_i)$ by bin (y-axis)



Common methods

Platt scaling

pass s_i through the sigmoid

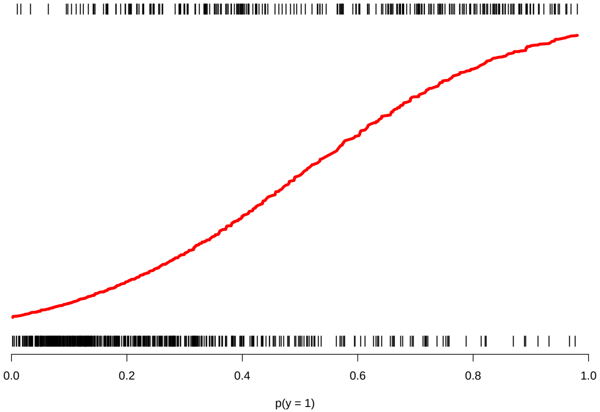
$$P(c_i|s_i) = \frac{1}{1 + \exp(As_i + B)}$$

where A and B are the solution to

$$\underset{A,B}{\operatorname{argmax}} - \sum_i y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

Platt scaling

applied to the Pima Indian Diabetes scores



Isotonic regression

a strictly-nondecreasing piecewise linear function m, where

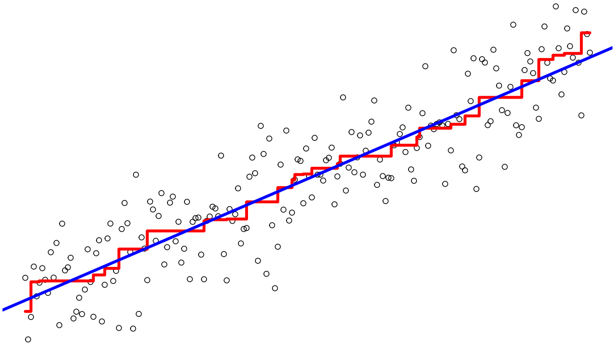
$$y_i = m(s_i) + \epsilon$$

fit such that

$$\hat{m} = argmin_z \sum_i (y_i - z(s_i))^2$$

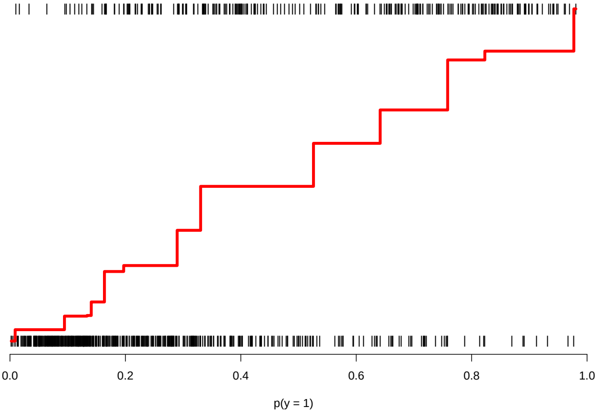
Isotonic regression

linear and isotonic regression fit to random noise with drift



Isotonic regression

applied to the Pima Indian Diabetes scores

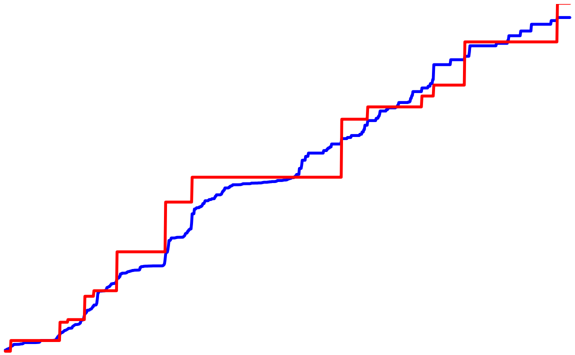


Notes on applying calibration

- it's really easy to overfit
  - calibration partition
  - cross-validation
- isotonic regression is generally more flexible (and can closely approximate sigmoid)
- best technique is dependent on the family of model used to generate s_i

Bootstrap aggregated isotonic regression

make it smoother by aggregating and averaging over 1000 resampled isotonic regression fits

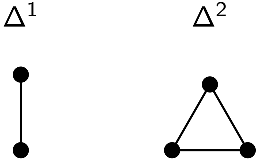


Extensions to $k > 2$

Probabilistic classification as a simplex

- if we view the task of probabilistic classification as a vector-valued function, we can visualize the co-domain of this task as assigning the location of a prediction in a regular (unit) simplex, Δ^{K-1}
- why is this hard when $K > 2$?

Probabilistic classification as a simplex



trivial with Δ^1 because we're only concerned with one unknown value and its complement. With $\Delta^{K>2}$ the simplex becomes a triangle, tetrahedron, five-cell, etc.

Multi-class probability estimation

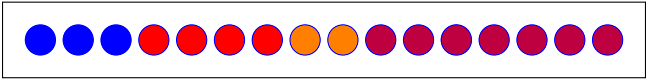


Figure 1: classification problem with $k = 4$

Strategy: decompose into separate binary classification problems

- one vs. all
- all pairs

One vs. all

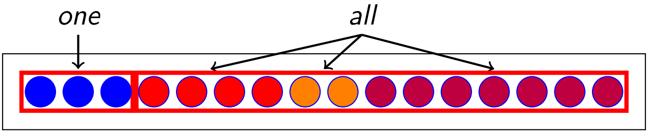


Figure 2: *one vs. all* reduces to $k - 1$ calibrations

All pairs

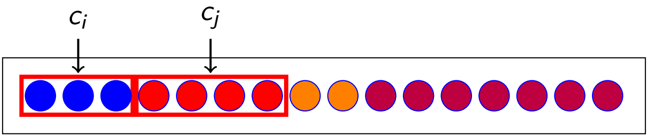


Figure 3: *all pairs* reduces to $\binom{K}{2}$ calibrations

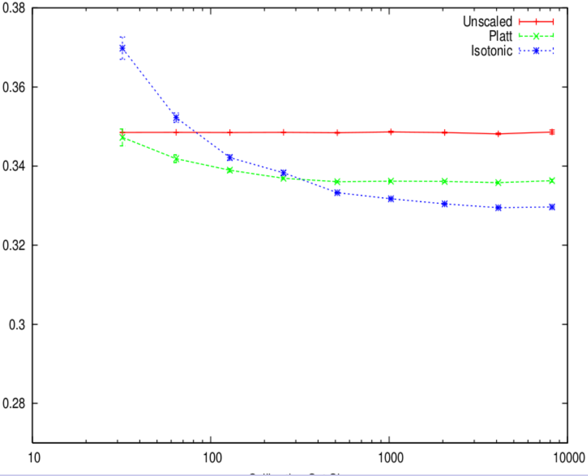
Combining multi-class probability estimates

- least squares: minimize squared error loss w/ non-negativity
- coupling (only *all pairs*): minimize log loss w/ non-negativity
- normalization (only *one vs all*): divide by sum of probabilities estimates

Experimental results

effect of calibration set size

from Niculescu-Mizil and Caruana, 2005



effect of multi-class combination method

from Zadrozny and Elkan, 2002

Method	MSE	Error Rate
NB Normalization	0.0326	0.1672
NB Least-Squares	0.0319	0.1672
NB Coupling	0.0304	0.1715
PAV NB Normalization	0.0241	0.1498
PAV NB Least-Squares	0.0260	0.1498
PAV NB Coupling	0.0260	0.1512
BNB Normalization	0.0163	0.0963
BNB Least-Squares	0.0164	0.0958
BNB Coupling	0.0160	0.1023
PAV BNB Normalization	0.0150	0.0946
PAV BNB Least-Squares	0.0150	0.0946
PAV BNB Coupling	0.0149	0.0935

boosting causes calibration issues

from Niculescu-Mizil and Caruana, 2005

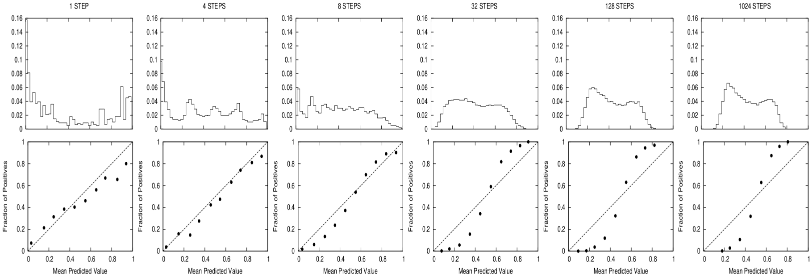


Figure 6: combination

Conclusion

References

- **Multivariate calibration of classifier scores into the probability space** by Martin Gebel
- **Transforming Classifier Scores into Accurate Multiclass Probability Estimates** by Zadrozny and Elkan
- **Obtaining Calibrated Probabilities from Boosting** by Niculescu-Mizil and Caruana
- **Predicting Good Probabilities With Supervised Learning** by Niculescu-Mizil and Caruana