

Probability calibration methodologies with local expert ensembles

CPT Nick Normandin

Introduction

What should you know about this brief?

- ▶ **Please ask questions as I go**

What should you know about this brief?

- ▶ **Please ask questions as I go**
- ▶ I've assumed some audience proficiency in modern machine learning techniques, but I will alternatively try to provide a heuristic understanding of the concepts presented

What should you know about this brief?

- ▶ **Please ask questions as I go**
- ▶ I've assumed some audience proficiency in modern machine learning techniques, but I will alternatively try to provide a heuristic understanding of the concepts presented
- ▶ Full accompanying R code is available

What should you know about this brief?

- ▶ **Please ask questions as I go**
- ▶ I've assumed some audience proficiency in modern machine learning techniques, but I will alternatively try to provide a heuristic understanding of the concepts presented
- ▶ Full accompanying R code is available
- ▶ This work was funded by the Omar N. Bradley Officer Research Fellowship in Mathematics

What is local expert?

I created a new kind of ensemble forecasting method called local expert regression. It involves the decomposition of a supervised learning task with a continuous target variable (*regression*) into a series of many $\{0, 1\}$ mappings corresponding to separate *binary probabilistic classification* tasks that produce estimates on the $[0, 1]$ interval.

Why is this useful ?!

Because you can aggregate the ensemble predictions to form a completely unique *probability distribution function* for each prediction. You can understand **risk** not just in terms of a model, but in terms of each individual forecast.

... see github.com/nnormandin/localexpert

What problem am I solving?

Most classification methods produce scores for class membership which are interpreted as measures of class affiliation probability. This is the foundation of local expert regression. However, these 'probabilities' are not usually **well-calibrated**.

Definition:

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.'

How do I propose to solve it?

If probabilities aren't properly calibrated, the PDFs interpolated from them won't be reliable. How can we deal with this?

1. Change the loss function

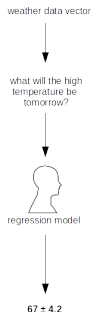
- ▶ $n^{-1} \sum_{i=1}^n -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$

2. Calibrate probabilities

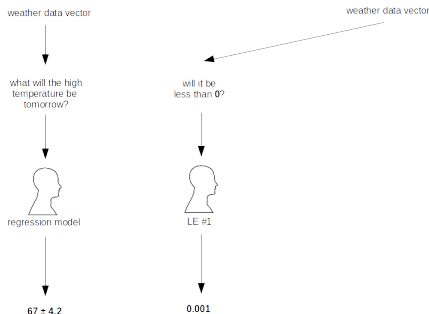
- ▶ isotonic regression, sigmoid transforms?

Local expert

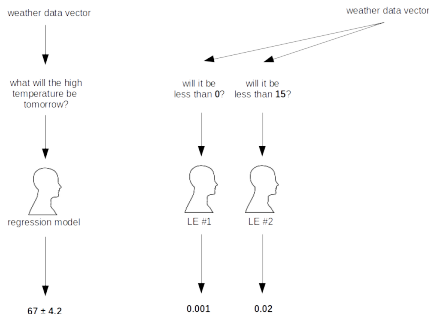
How is local expert different from normal regression?



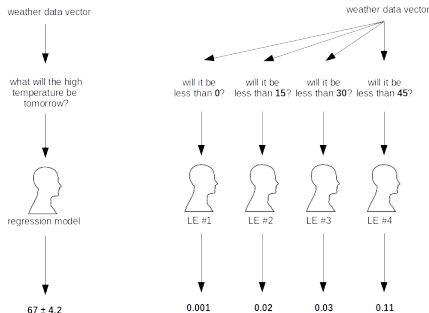
How is local expert different from normal regression?



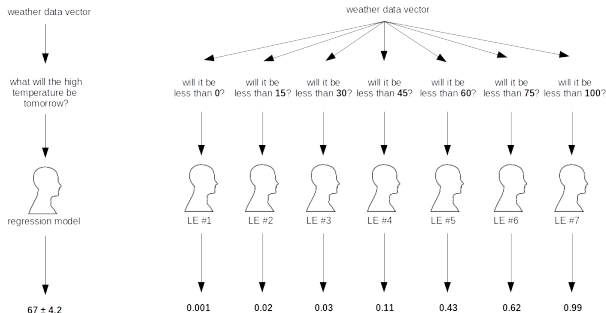
How is local expert different from normal regression?



How is local expert different from normal regression?



How is local expert different from normal regression?



How is local expert different from normal regression?

- ▶ The local expert predictions can be reconstructed to form an empirical CDF
- ▶ By fitting a spline to these points and then differentiating, we can create a unique PDF for each prediction
- ▶ We can forecast \hat{y} based on the moments of the distribution, or fit another meta-model on the output of the local expert ensemble (see Wolpert's work re: stacked generalization)
- ▶ **BUT**, all of these cool things we could do are dependent on having well-calibrated probability scores

Probability calibration

Why are some model scores poorly calibrated?

s_i dense around 0.5

- ▶ Maximal margin hyperplanes push scores away from extremes of distribution
- ▶ Common in support vector machines, boosted learners

s_i dense around 0, 1

- ▶ Model assumptions make class probabilities unrealistically confident
- ▶ Naive Bayes!

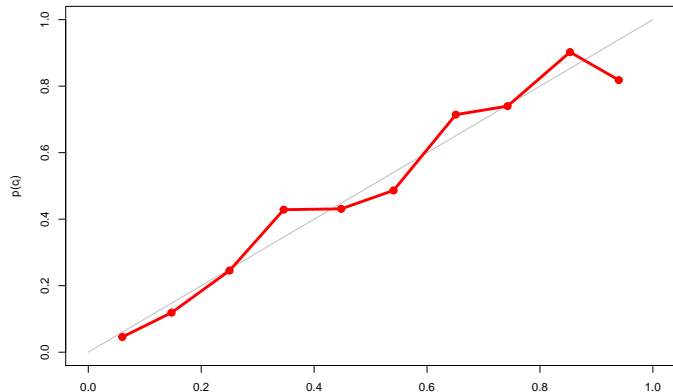
How can we visualize calibration?

Cross-validated class probabilities from a naive bayes model trained on the Pima Indian Diabetes data



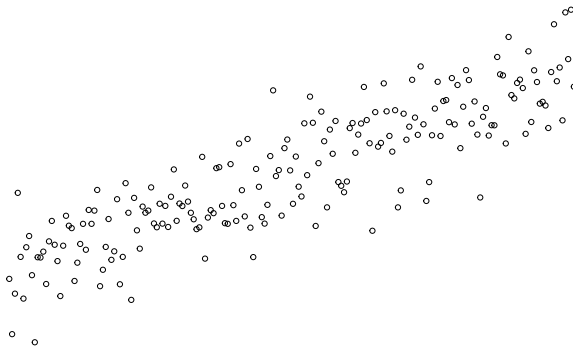
How can we visualize calibration?

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



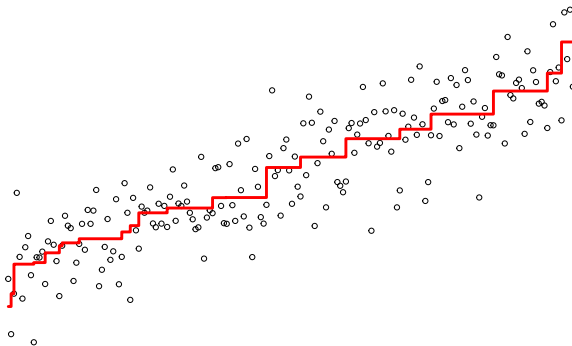
Method 1: Isotonic Regression

A strictly-nondecreasing piecewise linear function m , where $y_i = m(s_i) + \epsilon$ fit such that $\hat{m} = \operatorname{argmin}_z \sum_i y_i - z(s_i)^2$.



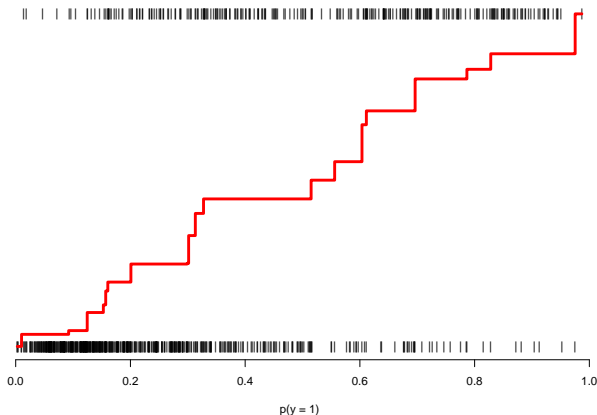
Method 1: Isotonic Regression

A strictly-nondecreasing piecewise linear function m , where $y_i = m(s_i) + \epsilon$ fit such that $\hat{m} = \operatorname{argmin}_z \sum_i y_i - z(s_i)^2$.



Method 1: Isotonic Regression

Applying it to the Pima Indian Diabetes estimates from earlier



Method 2: Platt Scaling

Pass s_i through the sigmoid

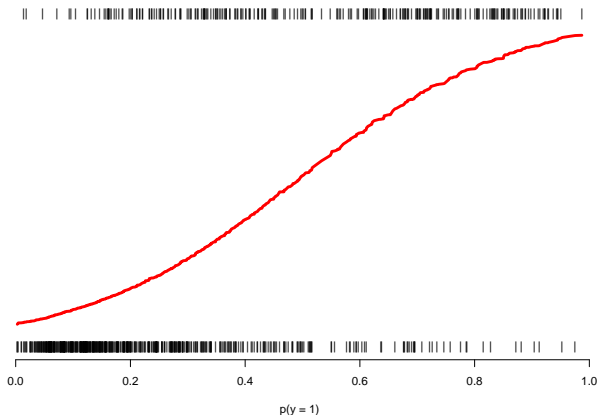
$$P(c_i | s_i) = \frac{1}{1 + \exp(As_i + B)}$$

where A and B are the solution to

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

Method 2: Platt Scaling

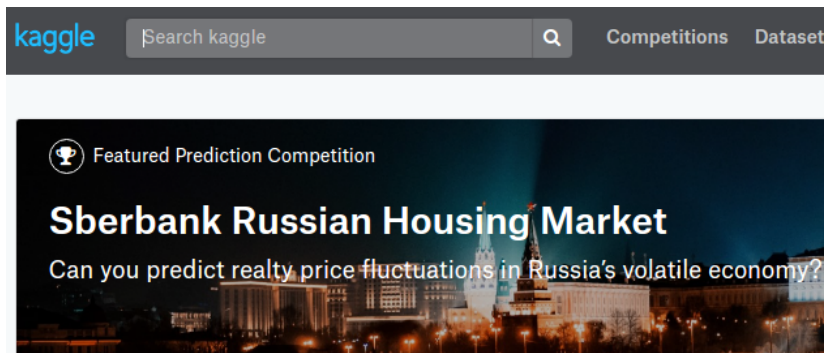
Applying it to the Pima Indian Diabetes estimates from earlier



Case study

The task

- ▶ 30,000 records of 300+ variables related to Russian housing transactions
- ▶ very dirty, lots of multicollinearity



Preparing the data

I cleaned and pre-processed separately, so we'll just read in those files and partition the train and test sets.

```
X <- readRDS('./dataX')
y <- readRDS('./dataY')

trainset <- createDataPartition(y, p = 0.7)[[1]]

Xtrain <- X[trainset,]; Xtest <- X[-trainset,]
ytrain <- y[trainset]; ytest <- y[-trainset]
```

COA 0: Tune and train a regression model

```
mod0 <- train(x = Xtrain, y = ytrain, method = 'gbm',  
              tuneLength = 5, trControl = tc,  
              verbose = FALSE)  
  
mod0_preds <- ExtractBestPreds(mod0)  
mod0_preds[mod0_preds < 0] <- 0
```

COA 1A: Use a local expert ensemble

Map the continuous y vector into a binary matrix

```
yb <- BinCols(ytrain, n = 25)
```

Induce separate models across each column in the matrix

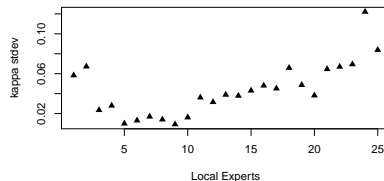
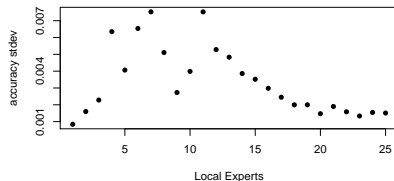
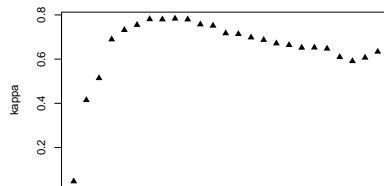
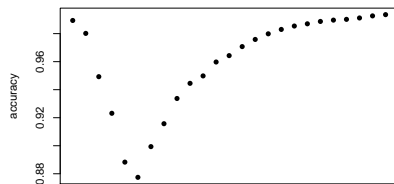
```
LEs <- TrainLEs(x = Xtrain, bincols = yb$cols, trControl = tc,  
                method = 'gbm', n.folds = 8, tuneLength = 6,  
                verbose = FALSE)
```

```
##      user      system   elapsed  
## 71152.460   112.044 12061.703
```

```
LE_info <- ExtractModelInfo(LEs)
```

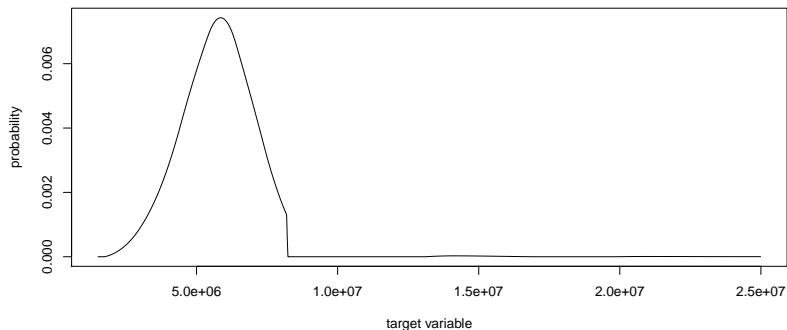
COA 1A: Use a local expert ensemble

```
sink('/dev/null'); PlotLEs(LE_info); sink()
```



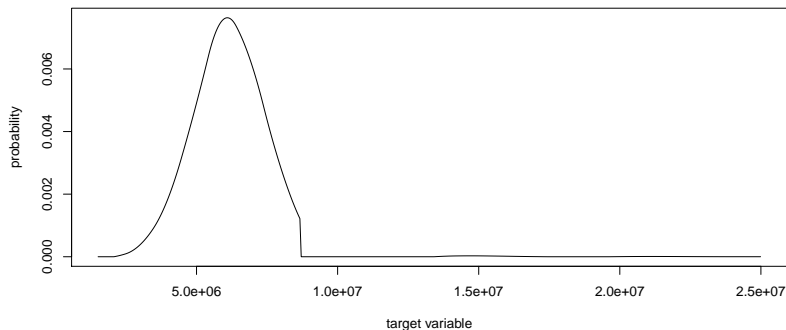
COA 1A: Use a local expert ensemble

Now for each separate instance, we're predicting a distribution instead of a value:



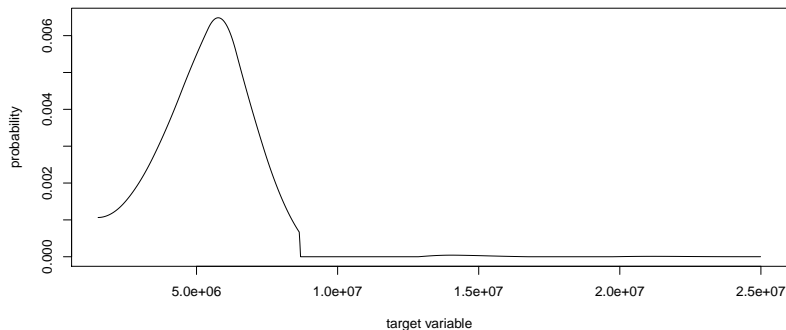
COA 1A: Use a local expert ensemble

Now for each separate instance, we're predicting a distribution instead of a value:



COA 1A: Use a local expert ensemble

Now for each separate instance, we're predicting a distribution instead of a value:



COA 1A: Use a local expert ensemble

Predict \hat{y} using just the means of the empirical distributions

```
LE_fits <- FitMatrix(LE_info$preds.matrix, yb$y.vals)
```

COA 1B:

Fit a stacked meta-model on the local expert layer output (include distribution mean as additional meta-feature).

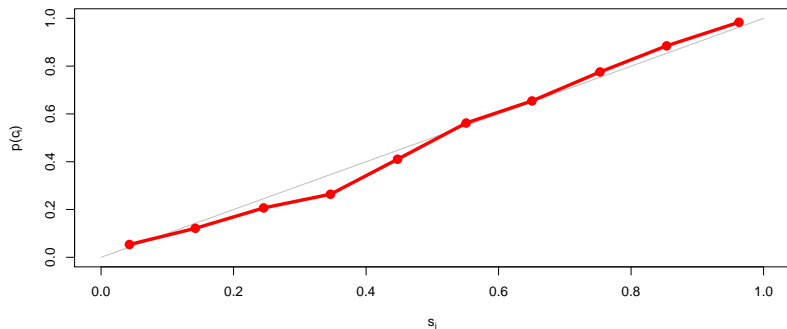
```
meta_X1 <- pm  
meta_X1$mean <- LE_fits$mean
```

```
## Warning in `[<-.data.table`(x, j = name, value = value): Supplied 14  
## items to be assigned to 20171 items of column 'mean' (recycled leavi  
## remainder of 1399 items).
```

```
mod_1B <- train(x= meta_X1, y = ytrain, method = 'gbm',  
               tuneLength = 10, trControl = tc1,  
               verbose = FALSE)
```

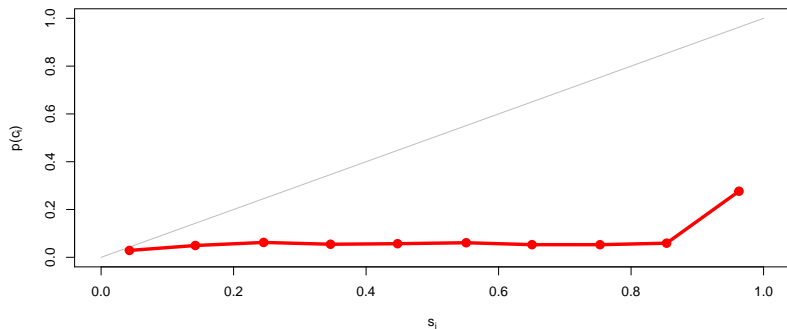
COA 2A: Local expert with calibrated probabilities

The output of each local expert must be calibrated independently. Each has a distinct reliability plot, and class imbalance has a substantial effect on the probability scores generated.



COA 2A: Local expert with calibrated probabilities

The output of each local expert must be calibrated independently. Each has a distinct reliability plot, and class imbalance has a substantial effect on the probability scores generated.



COA 2A: Local expert with calibrated probabilities

Instead of simply fitting an isotonic regression model to our local expert output, we can remove some of the bumpiness of the step function and reduce overfitting by bootstrapping. The steps are:

1. Draw random sample with replacement
2. Fit isotonic step function using PAVA algorithm
3. Repeat steps 1 & 2 for 500 iterations
4. Average all step functions
5. Return final 'step function' (still monotonic)

COA 2B: Stacked model with calibrated probabilities

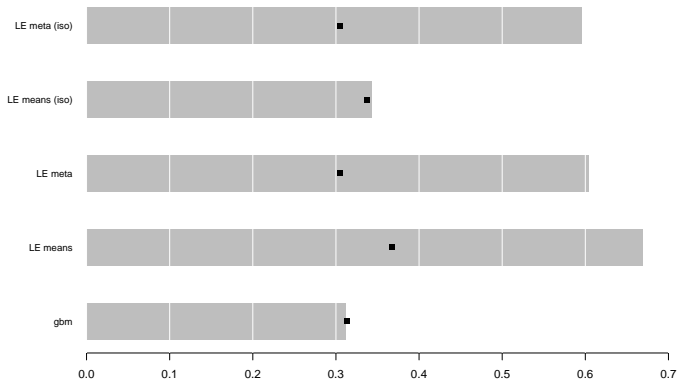
Fit a stacked meta-model on the local expert layer output (include distribution mean as additional meta-feature).

```
meta_X2 <- pm2
meta_X2$mean <- LE_fits2$mean

mod_2B <- train(x= meta_X2, y = ytrain, method = 'gbm',
                tuneLength = 8, trControl = tc1,
                verbose = FALSE)
```


Results

Model CV and out-of-sample RMLSE



Conclusion

Which topics require more research?

1. Compensation for class imbalance (SMOTE?)
2. Kappa-based optimization methods
3. High-level parallelization

Questions