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 that I've called local expert regression. It involves the decomposition of a supervised learning task with a continuous target variable (*regression*) into a series of 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 interpereted 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.'

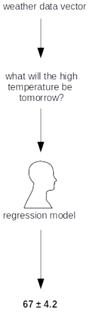
How do I propose to solve it?

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

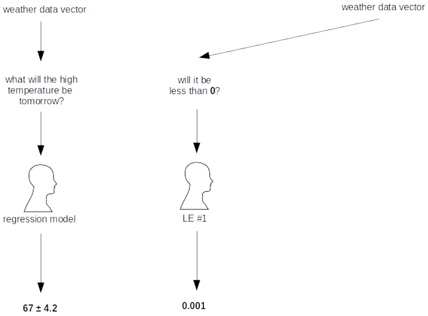
1. Change the loss function
   - $n^{-1}\sum_{i=1}^{n} -y_i \log(p_i) - (1-y_i)\log(1-p_i)$
2. Calibrate probabilities
   - isotonic regression, sigmoid transforms?

Local expert

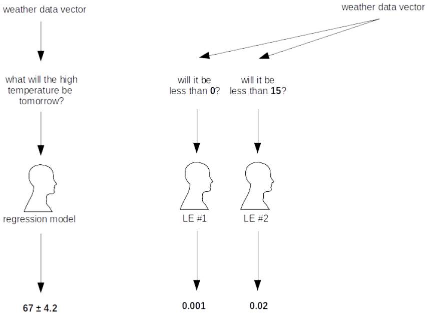
How is local expert different from normal regression?



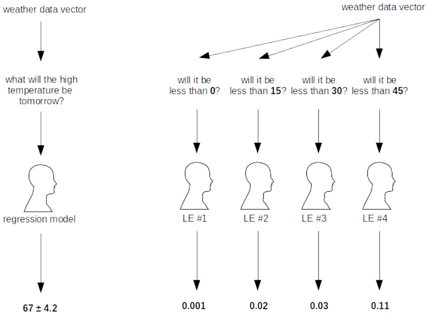
How is local expert different from normal regression?



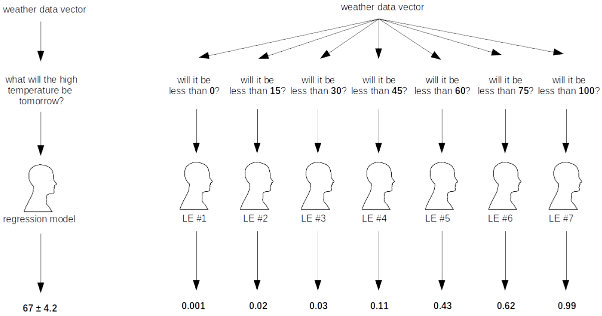
How is local expert different from normal regression?



How is local expert different from normal regression?



How is local expert different from normal regression?



How is local expert different from normal regression?

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

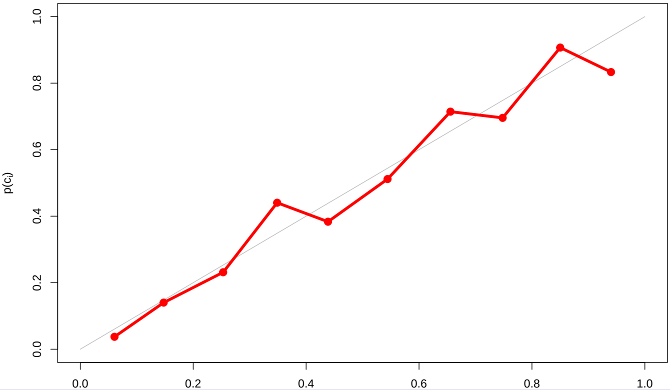
- Model assumptions make class probabilites unrealistically confident
- Naive Bayes!

How can we visualize calibration?

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

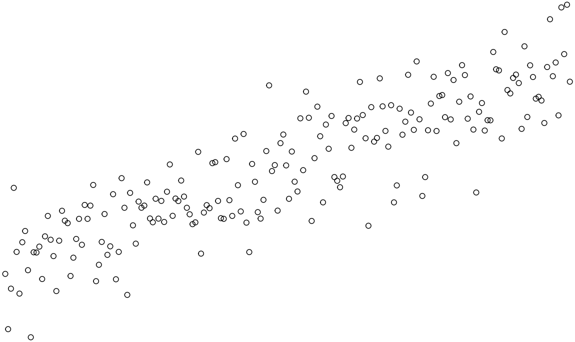
How can we visualize calibration?

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



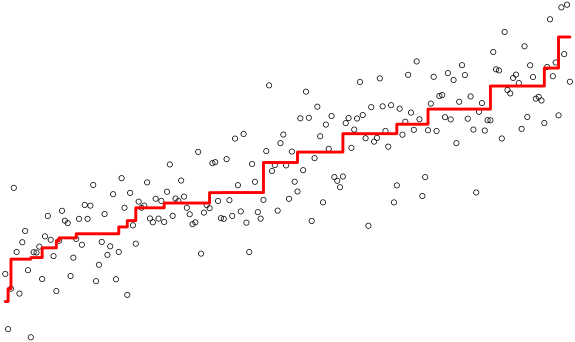
Method 1: Isotonic Regression

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



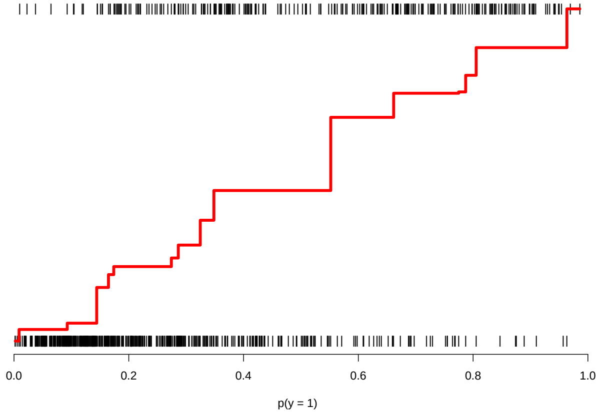
Method 1: Isotonic Regression

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



Method 1: Isotonic Regression

Applying it to the Pima Indian Diabetes estimates from earlier



Method 2: Platt Scaling

Pass s_i through the sigmoid

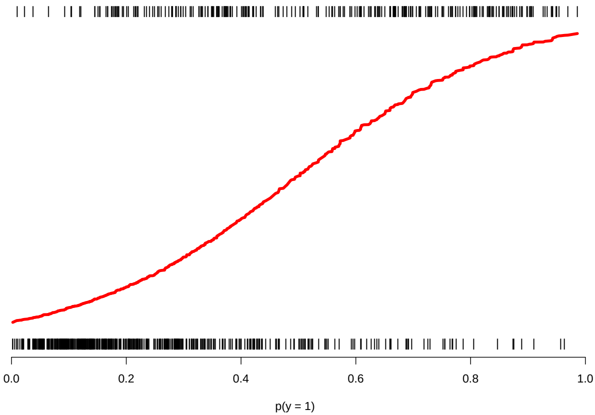
$$P(c_i|s_i) = \frac{1}{1 + \exp(As_i + B)}$$

where A and B are the solution to

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

Method 2: Platt Scaling

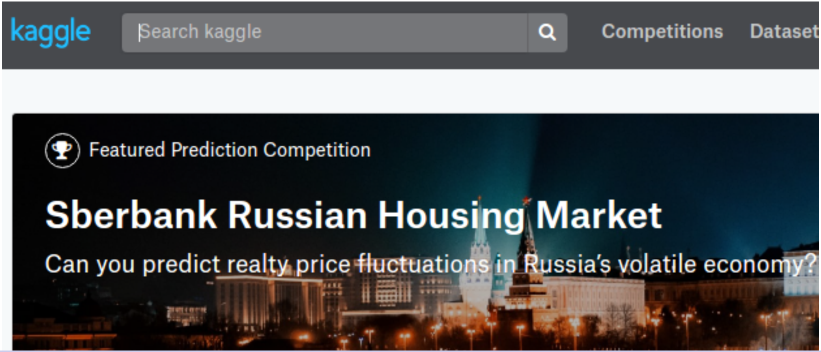
Applying it to the Pima Indian Diabetes estimates from earlier



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 = 0.5)[[1]]

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

COA 0: Tune and train a regression model

COA 1: Use a local expert ensemble

```r
library(localexpeRt)
yb <- BinCols(y, n = 30)
```

COA 3: Local expert with calibrated probabilities

Results

Conclusion

What have I demonstrated?

Which topics require more research?

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

Questions