Homework 2

36-708, Spring 2023

Due March 17 at 5PM EST

Please attach all code to your homework. In an RMarkdown document for example, this can be done in one line (see Yihui Xie's website for how to do this).

1 Taking a closer look at stacking weights (25pt)

In this question, we consider a model ensemble by weighting the outputs of different models and obtain the optimal set of weights for minimizing the squared risk. Consider the following setup.

- Data model: $(X, Y) \sim P_{X,Y}$ where $P_{X,Y}$ is a distribution on $\mathbb{R}^d \times \mathbb{R}$ with the regression function denoted by $f^\star(x) := \mathbb{E}(Y \mid X = x)$.
- Individual learners: $f_1, f_2, \ldots, f_T$ each mapping $\mathbb{R}^d \to \mathbb{R}$.
- Stacked learner: $f = \sum_{i=1}^T w_i f_i$.

First, we obtain a certain conditions on weights to keep the final prediction of the weighted model within certain reasonable limits. Consider the following requirement on the final prediction.

- Constrained output: Suppose we require that at any input $x \in \mathbb{R}^d$, the final prediction of the weighted learner be within the maximum and minimum of the predictions of all the individual learner predictions, i.e., $f_{\min}(x) \le f(x) \le f_{\max}(x)$, where $f_{\min}(x) = \min\{f_1(x), \ldots, f_T(x)\}$ and $f_{\max}(x) = \max\{f_1(x), \ldots, f_T(x)\}$.

(a) Show that

$$f_{\min}(x) \le f(x) \le f_{\max}(x) \text{ for any } f_1, \ldots, f_T \iff w_i \ge 0 \text{ for each } i \text{ and } \sum_{i=1}^T w_i = 1.$$

1.1 Optimizing the weights subject to the constraints

Next, subject to the simplified sum constraint on the weights (that they need to add to 1), we find the best possible set of weights to minimize the squared risk as defined below.

- Metric: squared risk of f defined as $\mathbb{E}[(f(X) - f^\star(X))^2]$

(a) Show that the squared risk of f can be written as $\sum_{i=1}^T \sum_{j=1}^T w_i C_{ij} w_j$, where $C_{ij} = \mathbb{E}[(f_i(X) - f^\star(X))(f_j(X) - f^\star(X))]$ are certain model correlations.

(b) Define the matrix $\mathbf{C} \in \mathbb{R}^{T \times T}$ by $\mathbf{C} = (C_{ij})$. Assume $\mathbf{C}$ is invertible. Show that subject to the constraints on weights that $\sum_{i=1}^T w_i = 1$, the optimum weights to minimize the squared risk are given by:

$$w_i = \frac{\sum_{j=1}^T \mathbf{C}_{ij}^{-1}}{\sum_{k=1}^T \sum_{j=1}^T \mathbf{C}_{kj}^{-1}}$$

Hint: Use the method of Lagrange multipliers.

(c) Interpret the optimum weights. In particular, comment on the cases when the weights would be more symmetric and when the weights would be more asymmetric. How do the weights simplify if the model correlations, C_{ij} for $i \neq j$, are all 0? Interpret the simplified weights.

2 Stacking versus best model in action (25pt)

In the previous question, we obtained an optimal set of weights for combining models subject to certain reasonable constraints on the weights. But, in practice, we do no have access to the model correlations as defined in the previous question. Thus, we have to somehow learn the weights as well. Since both the models and the model weights are to be learned from the same data, we need to penalize the complexities of individual models somehow. One way to do is by considering the cross-validated prediction for different models. We then find an optimal set of weights to minimize the empirical squared error subject to certain weight constraints.

In this question, we consider such (stacked) model combination with individual models and see if stacking adds any benefit. Consider the following setup.

- Dataset $\{(x_i, y_i)\}_{i=1}^n$
- Individual learners: $f_1, \ldots, f_T : \mathbb{R}^d \to R$
- Stacked learner $f = \sum_{i=1}^T w_i f_i$
- Stacking weights $w_i, i = 1, \ldots, T$
- Weight constraints $w_i \geq 0, \sum_{i=1}^T w_i = 1$
- Stacking objective:

$$\frac{1}{V} \sum_{v=1}^{V} \sum_{j=1}^{n_v} \left(y_j^v - f^{-v}(x_j^v) \right)^2$$

  Here, $V = 10$ is the number of cross-validation folds, (x_j^v, y_j^v) is the j^{th} datapoint in fold v, and f^{-v} is the stacked learner trained on all of the data except for those in fold v, $(x_j^v, y_j^v)_{j=1}^{n_v}$.

2.1 Single best model

(a) Show that if we restrict the weights $w_i \in \{0, 1\}$ in addition to the above set of constraints, then the best combined learned is simply the individual learner with the smallest cross-validation error.

2.2 Stacked versus single best model

Next, we compare the single best model with best combined model for the following specific choices of dataset and individual learners.

- Dataset: Ames Housing Dataset with the following columns:
  - "Bldg_Type",
  - "Lot_Area",
  - "House_Style",
  - "Overall_Qual",
  - "Kitchen_Qual",
  - "Heating_QC",
  - "Sale_Condition",
  - "Year_Remod_Add",

  - "First_Flr_SF",
  - "TotRms_AbvGrd",
  - "Sale_Price"

  Your goal will be to predict `Sale_Price`. This dataset can be found in the R package, `AmesHousing`. Alternatively, the raw data can be found at

  http://jse.amstat.org/v19n3/decock/AmesHousing.txt with the documentation in

  http://jse.amstat.org/v19n3/decock/DataDocumentation.txt

- Train and test split: use 3:1 random split
- Individual learners (4 total):
  1. k-nearest neighbors with $k \in \{1, 6\}$.
  2. Linear regression.
  3. Random forests (you can use the default arguments provided in the software package you use). If using R, the `ranger` package is recommended for computational reasons.

(a) Train individual regressors and obtain cross-validated predictions to be used for stacking.

(b) Find the single best regressor among the regressors and its test error.

(c) Find the best stacking weights with the above set of weight constraints and construct the corresponding stacked regressor and obtain its test error. Compare it with the test error of the single best regressor.

(d) Repeat (a)–(c) *without* random forests. What do you notice about the stacked-versus-best MSE in both cases?

3 Deeper dive into random forests (25pt)

In this problem, we will use random forests for classification on a labeled spam database, https://archive.ics.uci.edu/ml/datasets/Spambase.

(a) Fit random-forest classifiers on the spam data. Try various suitable values of m, the number of predictor variables selected at random as candidates for splitting when growing the trees.

(b) Plot the train, out-of-bag, and test error of random forests with various values of m that you tried as a function of number of trees.

(c) Summarize your findings. In particular, comment on the following. How fast does the training error decrease? Does the training error reach zero? Does the testing error increase beyond a certain point?

4 Classifier calibration (25pt + 10pt bonus)

In this problem, we will calibrate classifiers including the random forest classifier from the previous problem. We will use the same spam data as in the previous problem.

Consider three separate classifiers:

- Random forest with $\sqrt{d}$ variables for splitting at a node
- Logistic regression
- The constant classifier which outputs 1 if the majority of labels are 1s and 0 otherwise.

Create a 70-30 (train+calibration)-test split. We will use the 'test' set for reliability diagrams.

(a) Use the first split of the data to train each classifier (do not attempt to calibrate the classifier output yet). Using each of the trained models and the test split of the data:

   (a) Apply each trained classifier to the test data.
   (b) Bin the test data into 10 bins with roughly equal proportions based on the classifier output.
   (c) Compute the average value of the classifier within each bin.
   (d) Compute the empirical probabilities (observed proportion of data labeled as spam) within each bin.

   Finally, plot the average value of the classifier (on the x axis) against the empirical probabilities (on the y axis) for each of the 10 bins. This plot is also called a 'reliability' diagram. Do any of the above classifiers already appear to be (close to) calibrated? Which models seem most in need of calibration?

(b) Now, use the first 500 observations from the first split for calibration and the rest for model training. Specifically:

   (a) Train each classifier using the first split, minus the first 500 observations.
   (b) Apply the model to the first 500 observations and note the classifier output.
   (c) Bin these 500 data points into 10 bins with roughly equal proportions based on the classifier output.
   (d) Compute the empirical probabilities (observed proportion of data labeled as spam) within each bin.

   When applying the model to the test dataset, first generate the classifier output. Then, match the classifier output to the appropriate bin and use the computed empirical probabilities (from the corresponding bin in the calibration set) as the predicted probability. Plot reliability diagrams using the test set.

(c) Comment on the above plots. In particular, do the classifiers seem calibrated? How does the sharpness compare among these three?

(d) **(Bonus, 10pt)** Other methods for calibration include *Platt scaling* and *isotonic regression*. In this question, we will explore how these methods compare with binning.

   (a) **Platt scaling**: Platt scaling uses logistic regression to calibrate the classifier output. Specifically, for a trained classifier f and a new set of data points to be used for calibration, which we denote $\{(x_i, y_i)\}_{i\in(1,500)}$ in the context of this problem, we:
      i. Train a logistic model, where
      $$\mathbb{P}(y_i = 1|f(x_i)) = \frac{1}{1 + e^{-\alpha+\beta f(x_i)}},$$
      and the unknown parameters $\hat{\alpha}$ and $\hat{\beta}$ are estimated from the calibration data.
      ii. Denote $g(x_i) := \frac{1}{1+e^{-\hat{\alpha}+\hat{\beta}f(x_i)}}$. In words, $g(x_i)$ is the classifier f applied to a particular data point x_i and then adjusted using the logistic model.
      iii. For a new test data point outside of both the training and calibration datasets, x_{test}, we model $\mathbb{P}(y_{\text{test}} = 1) = g(x_{\text{test}})$.

   Apply this procedure to the spam dataset and plot reliability diagrams using the test dataset again.

   (b) **Isotonic regression**. When using binning, you may have noticed that the target probabilities are no longer monotonic with respect to the output of each classifier. Isotonic regression is an approach that attempts to find a line of best fit under the constraint that the output of the regression must be monotonic. In particular:

i. Train each classifier using the first split, minus the first 500 observations.

ii. Apply the model to the first 500 observations and note the classifier output.

iii. Bin these 500 data points into 10 bins with roughly equal proportions based on the classifier output.

iv. Compute the empirical probabilities (observed proportion of data labeled as spam) within each bin.

v. Train an isotonic regression on the empirical probabilities generated above. For R users, this can be done using the PAVA function in the ISO library. Python users may wish to use the isotonic regression function in SKLEARN.

When applying the model to the test dataset, first generate the classifier output. Then, match the classifier output to the appropriate bin but use the isotonic regression output from the last step as the predicted probability. Again, plot reliability diagrams for this procedure using the test dataset.

(c) Comment on the above outputs. What are the advantages and disadvantages of each of the three calibration methods? Which would be your preferred approach to calibration for this problem?