R Assignment 4 - Super Learner

Laura B. Balzer

Biostat 683 - Intro. to Causal Inference

Assigned: November 10, 2021
Write-ups due: Uploaded to your personal GoogleDrive folder by November 23, 2021 by 11:59pm. Please answer all questions and include relevant R code. You are encouraged to discuss the assignment in groups, but should not copy code or interpretations verbatim. Use of RMarkdown is strongly encouraged.

1 Background Story

Given the success of our previous studies, we have been hired by lead NGOs to build a prediction function for malnutrition at the community-level. The outcome of interest Y is the average mid-upper arm circumference (MUAC) of children aged 6-59 months in each community. In this age range, a MUAC less than 110 mm indicates severe acute malnutrition. (Other indicators of severe acute malnutrition include visible severe wasting, nutritional edema, and a standardized weight for height lower than 3 standard deviations from the median.)

Figure 1: `http://www.doctorswithoutborders.co.nz/education/activities/braceletoflife/index.html`

We have data on the following community-level predictor variables:

- W1 - community's access to potable water (1-yes; 0-no)
- W2 - whether the community is located in a stable region (1-yes; 0-no)
- W3 - a measure of the community's socioeconomic status (on a scale from 0-5)
- W4 - the proportion of children visiting a health center in the last year for common childhood illnesses (e.g., diarrhea and pneumonia)
- W5 - the number of health facilities or therapeutic feeding centers in a community (min=1, max=4)

Let $W = \{W1, W2, W3, W4, W5\}$ be the set of predictors.

2 Import and explore the data set `RAssign4.SL.csv`.

1. **Use the `read.csv` function to import the data set and assign it to data frame `ObsData`.**
2. **Use the `names`, `tail` and `summary` functions to explore the data.**

3. **Use the `nrow` function to count the number of communities in the data set. Assign this number as `n`.**

3 Code discrete Super Learner to select the estimator with the lowest cross-validated risk estimate.

The first step of the Super Learner algorithm (and more generally loss-based learning) is to define the target parameter $\bar{Q}_0(W) = \mathbb{E}_0(Y|W)$ as the minimizer of the expectation of a loss function:

$$\bar{Q}_0(W) = argmin_{\bar{Q}}\ \mathbb{E}_0\big[L(O, \bar{Q})\big]$$

We will use the L2 loss function:

$$L(O, \bar{Q}) = (Y - \bar{Q}(W))^2$$

but note many other loss functions are possible. The expectation of the loss function is called the *risk*. The second step is to define a library of candidate estimators. Suppose that before beginning the analysis we talked to subject matter experts and came up with the following candidate estimators for the conditional expectation of MUAC, given the predictors:

$$\begin{aligned}
\bar{Q}^a(W) &= \beta_0 \\
\bar{Q}^b(W) &= \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W3 + \beta_4 W4 + \beta_5 W5 \\
\bar{Q}^c(W) &= \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W3 + \beta_4 W4 + \beta_5 W5 + \beta_6 W2^* W5 \\
\bar{Q}^d(W) &= \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W3 + \beta_4 W4 + \beta_5 W5 + \beta_6 W1^* W3 + \beta_7 W2^* W5
\end{aligned}$$

Therefore, our library consists of 4 parametric regressions, denoted with the superscripts $a - d$. The first algorithm corresponds to the simple mean and the second to main terms regression. The third and fourth algorithms include key interaction terms. We will choose the candidate estimator with the smallest cross-validated risk estimate. In other words, we are going to select the estimator with the lowest cross-validated mean squared prediction error.

1. **Briefly discuss the motivation for using discrete Super Learner (a.k.a., the cross-validation selector).**

2. **Set the seed to 1, and then split the data into $V = 20$ folds.** With $n = 500$ observations total, we want $n/20 = 25$ observations in each fold.

3. **Create a matrix `Pred` with 500 rows for the communities and 4 columns to hold the cross-validated predictions for each community according to each candidate algorithm.**

4. **Create an empty matrix `CV.risk` with 20 rows and 4 columns for each algorithm, evaluated at each fold.**

5. **Use a `for` loop to fit each estimator on the training set (19/20 of the data); predict the expected MUAC for the communities in the validation set (1/20 of the data), and evaluate the cross-validated risk.**

   (a) **Since each fold needs to serve as the training set, have the `for` loop run from `V` is 1 to 20.** First, the observations in $Fold = 1$ will serve as the validation set and other 475 observations as the training set. Then the observations in $Fold = 2$ will be the validation set and the other 475 observations as the training set... Finally, the observations in $Fold = 20$ will be the validation set and the other 475 observations as the training set.

   (b) **Create the validation set as a data frame `valid`, consisting of observations with `Fold` equal to `V`.**

(c) **Create the training set as a data frame `train`, consisting of observations with `Fold` not equal to `V`.**

(d) **Use `glm` to fit each algorithm on the training set. Be sure to specify `data=train`.**

(e) **For each algorithm, predict the average MUAC for each community in the validation set. Be sure to specify the `type='response'` and `newdata=valid`.**

(f) **Save the cross-validated predictions for each community in the validation set at the appropriate row in the matrix `Pred`.**

(g) **Estimate the cross-validated risk for each algorithm with the L2 loss function.** Take the average of the squared differences between the observed outcomes Y in the validation set and the predicted outcomes. **Assign the cross-validated risks as a row in the matrix `CV.risk`.**

6. **Select the algorithm with the lowest average cross-validated risk across the folds.** Hint: use the `colMeans` function.

7. **Fit the chosen algorithm on all the data.**

8. **How can we come up with an even better prediction function than the one selected?** (This question is not about improving computing in `R`, but about improving our ability to predict the outcome.)

4 Bonus: Completely Optional - Coding the weights

1. Write your own `R` code to estimate the optimal convex combination of weights using the L2 loss function. (In other words, do not use the `SuperLearner` package.)

2. Apply your weights to generate the ensemble based predictions.

5 Use the `SuperLearner` package to build the best combination of algorithms.

1. **Load the Super Learner package with the `library` function and set the seed to `252`.**

2. **Use the `source` function to load script file `Rassign4.Wrappers.R`, which includes the wrapper functions for the *a priori*-specified parametric regressions.** *Note:* `SL.glm.EstA` should give identical results to `SL.mean`; both are taking the empirical mean. Likewise, `SL.glm.EstB` should give identical results to `SL.glm`; both are running main terms regression.

3. **Specify the algorithms to be included in Super Learner's library.** Create a vector `SL.library` of the following algorithms:

```
> SL.library<- c('SL.glm.EstA', 'SL.glm.EstB', 'SL.glm.EstC', 'SL.glm.EstD',
+     'SL.ridge','SL.rpart', 'SL.earth')
```

   Here, we are expanding the library in two ways: (1) by including these new algorithms, and (2) by searching for the best convex combination of algorithms.
   Bonus: **Very briefly describe the algorithms corresponding to `SL.ridge`, `SL.rpart`, and `SL.earth`.**

4. **Create data frame `X` with the predictor variables.**

5. **Run the `SuperLearner` function. Be sure to specify the outcome `Y`, the predictors `X` and the library `SL.library`. Also include `cvControl=list(V=20)` in order to get 20-fold cross-validation.**

6. **Explain the output to relevant policy makers and stake-holders. What do the columns `Risk` and `Coef` mean? Are the cross-validated risks from `SuperLearner` close to those obtained by your code?**

7. **Briefly why we don't (or shouldn't) use Machine Learning-based predictions of $\mathbb{E}_0(Y|A,W)$ in a simple substitution estimator or Machine Learning-based predictions of $\mathbb{P}_0(A|W)$ in inverse probability weighting?**

6 Implement `CV.SuperLearner`

1. **Explain why we need `CV.SuperLearner`.**

2. **Run `CV.SuperLearner`.** Again be sure to specify the outcome `Y`, predictors `X`, and library `SL.library`. Specify the cross-validation scheme by including `cvControl=list(V=5))` and `innerCvControl=list(list(V=20))`. This might take a couple minutes.

```
> CV.SL.out<- CV.SuperLearner(Y=ObsData$Y, X=X, SL.library=SL.library,
+      cvControl=list(V=5), innerCvControl = list(list(V=20)) )
```

   This function is partitioning the data into V^*=5 folds, running the whole Super Learner algorithm in each training set (4/5 of the data), evaluating the performance on the corresponding validation set (1/5 of the data), and rotating through the folds. Each training set will itself be partitioned into V=20 folds in order to run `SuperLearner`.

3. **Explore the output.** For example, if the output object from `CV.SuperLearner` was `CV.SL.out`, run the following code.

```
> # summary of the output of CV.SuperLearner
> summary(CV.SL.out)
> #
> # returns the output for each call to Super Learner
> CV.SL.out$AllSL
> #
> # condensed version of the output from CV.SL.out$AllSL with only the coefficients
> # for each Super Learner run
> CV.SL.out$coef
> #
> # returns the algorithm with lowest CV risk (discrete Super Learner) at each step.
> CV.SL.out$whichDiscrete
```

   Only include the output from the *summary* function in your write-up, but *comment* on the other output.