R Lab 3 - SuperLearner!

Introduction to Causal Inference

Goals:

1. Import a data set into `R`.

2. Code discrete SuperLearner (a.k.a. the cross-validation selector) to select the estimator with the lowest cross-validated risk from a library of candidate algorithms.

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

4. Evaluate the performance of SuperLearner with `CV.SuperLearner`.

Next lab:

We will implement the inverse probability of treatment weighted (IPTW) estimator and explore the impact of positivity violations on estimator performance.

1 Background - Pirate Prediction!

You have been hired by the Queen to build the best predictor of a pirate's ship success at finding buried treasure Y. Your findings are the preliminary step in a potential intervention to increase the number ships returning with treasure. The Queen has paid a handsome fee to the harbor master at Tortuga for his data. Specifically, you have data on the following predictor variables:

- W1 - the possession of a treasure map (yes:1; no:0)
- W2 - the ship's supply of rum when returning to port (in gallons)
- W3 - the number of mutinies in the last year (min:0; max: 5 or more)
- W4 - the proportion of ship members who own a parrot, have a peg leg, and/or have an eye patch

Let $W = \{W1, W2, W3, W4\}$ represent the vector of these predictors.

Image 1: http://www.jokeroo.com/pictures/dogs/971264.html

Image2: http://www.charlierandallpetfoundation.org/pet-tips/keep-your-pet-safe-on-halloween/attachment/pirate-dog-costume2/

2   Import the data set `RLab3.SuperLearner.csv`

1. **Use the `read.csv` function to read in the data set and assign it to object `ObsData`.**

   > `ObsData<- read.csv("RLab3.SuperLearner.csv")`

   For this to work, the `RLab3.SuperLearner.csv` file needs to be in your working directory, which can be accessed with the `getwd()` function. Alternatively, you can tell `R` the full path of the file. For example, if the `RLab3.SuperLearner.csv` file was downloaded to my desktop, I could use the following command:

   > `ObsData<- read.csv("/Users/laura/Desktop/RLab3.SuperLearner.csv")`

2. **Use the `names`, `head` and `summary` functions to explore the data.**

3. **Use the `nrow` function to count the number of ships in the data set. Assign it to `n`.**

3   The curse of dimensionality!

Suppose your Queen believes that the only predictors of a ship's success are possession of a treasure map ($W1$) and the number of mutinies in the last year ($W3$). Since these are categorical characteristics, she demands that you estimate the conditional probability of finding treasure, given these characteristics $E_0(Y|W1, W3)$, with the non-parametric maximum likelihood estimator (NPMLE).

1. **Use the `table` function to count the number of ships within strata of map possession $W1$ and mutinies $W3$.**
   Hint: Columns in the data frame `ObsData` can be accessed with the `$` operator.

2. **Are you wary of estimating the conditional expectation of Y with the sample proportion in each strata (i.e. the NPMLE)?** For example, try estimating the conditional probability of finding treasure without a map and with five counts of mutiny: $E_0(Y|W1 = 0, W3 = 5)$.
   Hint: use the logical `&`.

   Take-home message: In many data applications, our statistical model $\mathcal{M}$ is non-parameteric, but the NPMLE is not well-defined. There can be strata with zero or only a few observations. In these situations, estimation of the conditional mean function with the NPMLE (i.e. a fully saturated parametric regression model) will be an overfit. Thereby, we need an alternative estimation approach. We could describe the probability of finding treasure given predictor variables with a lower dimensional parametric model. However, we often do not know enough to *a priori*-specify the correct parametric model for $E_0(Y|W1, W2, W3, W4)$. If the specified parametric model is incorrect, our point estimates and inference will be biased. Trying a bunch of parametric regressions, looking at the results, fiddling with the regrssion specifications, and choosing the "best"(using arbitrary criteria) also leads to biased point estimates and misleading inference. Therefore, we are going to use discrete SuperLearner to choose between a library of *a priori*-specified candidate prediction models.

4   Code discrete SuperLearner to select the estimator with the lowest cross-validated risk

The first step of the Super Learner algorithm (and more generally loss-based learning) is to define the target parameter $\bar{Q}_0(W) = E_0(Y|W)$ as the minimizer of the expectation of a loss function:

$$\bar{Q}_0(W) = argmin_{\bar{Q}} \ E_0\big[L(O, \bar{Q})\big]$$

Since the outcome is binary, we can use the L2 loss function or the negative log likelihood loss:

$$L(O, \bar{Q}) = (Y - \bar{Q}(W))^2$$
$$L(O, \bar{Q}) = -log[\bar{Q}(W)^Y(1 - \bar{Q}(W))^{1-Y}]$$

For the purposes of this lab, we are going to use the L2 loss function as our measure of performance. The expectation of the loss function under the true data generating distribution P_0 is called the *risk*.

The second step is to define a library of candidate estimators. Suppose that before prior to the analysis, pirate experts came up with the following candidate estimators to include in the library for the discrete SuperLearner (i.e. the cross-validation selector):

$$\bar{Q}_n^a(W) = expit[\beta_0 + \beta_1 W1 + \beta_2 W3 + \beta_3 W1^*W3 + \beta_5 W4^2]$$
$$\bar{Q}_n^b(W) = expit[\beta_0 + \beta_1 W1 + \beta_2 log(W2) + \beta_3 W3 + \beta_4 W4 + \beta_5 W3^*W4]$$
$$\bar{Q}_n^c(W) = expit[\beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W4 + \beta_4 W1^*W2 + \beta_5 W1^*W4 + \beta_6 W2^*W4 + \beta_7 W1^*W2^*W4]$$
$$\bar{Q}_n^d(W) = expit[\beta_0 + \beta_1 W1 + \beta_2 sin(W2^2) + \beta_3 W1^* sin(W2^2) + \beta_4 log(W4)]$$

where $W = (W1, W2, W3, W4)$ and *expit* is the inverse of the logistic function. Therefore, our library consists of four parametric models, denoted with the superscripts $a - d$. Using cross-validation, we can generate an "honest" estimate of risk for each candidate $\bar{Q}_n(A, W)$. We will choose the candidate estimator with the smallest cross-validated risk estimate. Here, we are going to select the estimator with the lowest cross-validated mean squared prediction error.

1. **Which algorithm do you think will do best at predicting the pirate ships success at finding treasure?**

2. **Create the following transformed variables and add them to data frame `ObsData` with the `data.frame` function.**

   ```
   > W2sq <- ObsData$W2*ObsData$W2
   > sinW2sq<- sin(W2sq)
   > logW2<- log(ObsData$W2)
   > W4sq <- ObsData$W4*ObsData$W4
   > sinW4sq <- sin(W4sq)
   > logW4 <- log(ObsData$W4)
   ```

3. **Split the data into $V = 10$ folds.** With $n = 1000$ observations total, we want $n/10 = 100$ in each fold. For simplicity let us define the first hundred observations to be the first fold, the second hundred to be the second fold, and so forth.

   (a) **Create the vector `Fold` by with the `c()` and `rep()` functions.**

      ```
      > Fold<- c(rep(1, 100), rep(2, 100), rep(3, 100),rep(4, 100),rep(5, 100),
      +   rep(6, 100),rep(7, 100), rep(8, 100), rep(9, 100), rep(10, 100))
      ```

      Alternatively, ou could use the `sample` function (without replacement) to get 10 folds of size 100.

   (b) **Add the `Fold` vector to the data frame `ObsData` with the `data.frame` function.**

4. **Create an empty matrix `CV.risk` with 10 rows and 4 columns to hold the cross-validated risk for each algorithm, evaluated at each fold.**

   Hint: the following code creates a matrix `temp` of `NA` of size 2 by 2:

   ```
   > temp<- matrix(NA, nrow=2, ncol=2)
   > temp

        [,1] [,2]
   [1,]   NA   NA
   [2,]   NA   NA
   ```

5. **To implement discrete SuperLearner, use a `for` loop to fit each estimator on the training set (9/10 of the data); predict the probability of finding treasure for the corresponding validation set (1/10 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 10.** First, the observations in `Fold=1` will serve as the validation set and other 900 observations as the training set. Then the observations in `Fold=2` will be the validation set and the other 900 observations as the training set... Finally, the observations in `Fold=10` will be the validation set and the other 900 observations as the training set.

   (b) **Create the validation set as a data frame `valid` consisting of observations with `Fold` equal to `V`.**
   Hint: Use the logical `==` to select the rows of `ObsData` with `Fold==V`.

   (c) **Create the training set as a data frame `train` consisting obserations with `Fold` not equal to `V`.**
   Hint: Use the logical `!=` to select the rows of `ObsData` with `Fold!=V`.

   (d) **Use `glm` to fit each algorithm on the training set.** Be sure to specify `family=` 'binomial' for the logistic regression and `data` as the training set `train`.

   (e) **For each algorithm, predict the probability of finding treasure for each ship in the validation set.** Be sure to specify the `type=`'response' and `newdata` as the validation set `valid`.

   (f) **Estimate the cross-validated risk estimate for each algorithm based on the L2 loss function.** Take the `mean` of the squared difference between the outcomes Y in the validation set and the predicted probabilities. **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.** Apply the `colMeans` function to the matrix `CV.risk`.

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

8. **Can we do better?**

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

1. **If you have not done so already, install the `SuperLearner` package.** Click on the `Packages & Data` tab. Select `Package Installer`. Search for `SuperLearner`. Check the box for `Install Dependencies` and click `Install Selected`.

2. **Load the SuperLearner package with the `library` function.**
   `> library('SuperLearner')`

3. **Read the help file on `SuperLearner`.**
   `> ?SuperLearner`

4. **The `SuperLearner` package uses wrapper functions. Use the `listWrappers()` function to see built-in candidate algorithms. For example, explore the wrapper function for stepwise regression `SL.step`.**

   `> SL.step`

5. **Use the `source` function to load script file `RLab3.SuperLearner.Wrappers.R`, which includes code for the wrapper functions for the *a priori*-specified parametric estimators.**

   `> source('RLab3.SuperLearner.Wrappers.R')`

6. **Specify the algorithms to be included in SuperLearner's library.** Create the following vector `SL.library` of character strings:

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

Note: We are choosing these simple algorithms in the interest of time. Remember Anna's Boy Band heuristic; there are a lot more fun and exciting algorithms out there. Feel free to try out others.

7. **Create data frame X with the predictor variables.** Include the original predictor variables as well as the transformed variables.
Hint: The following code uses the `subset` function to select all columns, but `Y`:

```
> X<- subset(ObsData, select= -Y )
```

8. **Run the `SuperLearner` function. Be sure to specify the outcome Y, the predictors X, the library `SL.library` and the `family`. Also include `cvControl=list(V=10)` in order to get 10-fold cross-validation.**

```
> SL.out<- SuperLearner(Y=ObsData$Y, X=X, SL.library=SL.library, family='binomial',
+     cvControl=list(V=10))
```

9. **Which algorithm had the lowest cross-validated risk? Which algorithm was given the largest weight when building the convex combination of algorithms? Are the cross-validated risks from `SuperLearner` close to those obtained by your code?**

6   Evaluate the performance of SuperLearner.

1. **Read the help file on `CV.SuperLearner`.**

```
> ?CV.SuperLearner
```

2. **Evaluate the performance of the SuperLearner algorithm by running `CV.SuperLearner`.** Again be sure to specify the outcome `Y`, predictors `X`, `family`, `SL.library`, `V=10` and `cvControl=list(V=10)`. This might take a couple minutes.

```
> CV.SL.out<- CV.SuperLearner(Y=ObsData$Y, X=X, SL.library=SL.library, family='binomial',
+     V=10, cvControl=list(V=10))
```

This function is partitioning the data into $V^*=10$ folds, running the whole SuperLearner algorithm in each training set (9/10 of the data), evaluating the performance on the corresponding validation set (1/10 of the data), and rotating through the folds. Each training set will itself be partitioned into $V=10$ folds in order to run `SuperLearner`.

3. **Explore the output with the `summary` function.**