R Assignment 3

PH252D Fall 2013
Introduction to Causal Inference

Assigned: October 28, 2013
Due: November 6, 2013
Write-up: 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. You need to bring your own completed assignment to class.

1 Background Story

Given the success of our previous studies, we have been hired by lead NGOs to build a prediction model for community-level severe acute malnutrition. 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 socio-economic 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 $W = \{W1, W2, W3, W4\}$ be the set of predictors.

2 Import and explore the data set `Rassign3.Fa2013.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 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:

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

Since the outcome is continuous and the target parameter is the conditional mean of the outcome Y given the covariates $W = (W1, W2, W3, W4, W5)$, we will use the L2 loss function:

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

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 you talked to subject matter experts and came up with the following candidate estimators for the conditional expectation of MUAC, given the covariates:

$$\begin{aligned}
\bar{Q}^a_n(W) &= \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 \text{sin}(W3) + \beta_4 W4^2 \\
\bar{Q}^b_n(W) &= \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W4 + \beta_4 \text{cos}(W5) \\
\bar{Q}^c_n(W) &= \beta_0 + \beta_1 W2 + \beta_2 W3 + \beta_3 W5 + \beta_4 W2^* W5 + \beta_5 W4^2 + \beta_6 \text{cos}(W5) \\
\bar{Q}^d_n(W) &= \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W5 + \beta_4 W1^* W2 + \beta_5 W1^* W5 + \beta_6 W2^* W5 + \beta_7 W1^* W2^* W5
\end{aligned}$$

Therefore, our library consists of four parametric models, denoted with the superscripts $a - d$. Finally, 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 SuperLearner (a.k.a. the cross-validation selector).**
2. **Create the following transformed variables and add them to data frame `ObsData`.**

```
> sinW3<- sin(ObsData$W3)
> W4sq <- ObsData$W4*ObsData$W4
> cosW5 <- cos(ObsData$W5)
```

3. **Split the data into $V = 20$ folds.** With $n = 5000$ observations total, we want $n/20 = 250$ observations in each fold. **Create the vector `Fold` and add it to the data frame `ObsData`.**
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 4750 observations as the training set. Then the observations in $Fold = 2$ will be the validation set and the other 4750 observations as the training set... Finally, the observations in $Fold = 20$ will be the validation set and the other 4750 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) **Estimate the cross-validated risk for each algorithm with the L2 loss function.** Take the `mean` 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.**
   Hint: use the `colMeans` function.

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

8. **Can we do better?**

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

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

2. **Use the `source` function to load script file `Rassign3.Fa2012.Wrappers.R`, which includes the wrapper functions for the *a priori*-specified parametric regressions.**

3. **Specify the algorithms to be included in SuperLearner'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.rpartPrune', 'SL.polymars', 'SL.mean')
```

   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.rpartPrune`, `SL.polymars` and `SL.mean`.**

4. **Create data frame `X` with the predictor variables.** Include the original predictor variables and the transformed 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?**

5 Implement `CV.SuperLearner`

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

2. **Run `CV.SuperLearner`.** Again be sure to specify the outcome `Y`, predictors `X`, library `SL.library`, folds `V=20` and `cvControl=list(V=20)`. This might take a couple minutes.

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

This function is partitioning the data into V*=20 folds, running the whole SuperLearner algorithm in each training set (19/20 of the data), evaluating the performance on the corresponding validation set (1/20 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 SuperLearner
> CV.SL.out$AllSL
> #
> # condensed version of the output from CV.SL.out$AllSL with only the coefficients
> # for each SuperLearner run
> CV.SL.out$coef
> #
> # returns the algorithm with lowest CV risk (discrete SuperLearner) at each step.
> CV.SL.out$whichDiscrete
```

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

6 Bonus! -Completely Optional

1. **Try adding more algorithms to the `SuperLearner` library.**
2. **Try writing your own wrapper function. The following code will give you a template.**

```
> SL.template

function (Y, X, newX, family, obsWeights, id, ...)
{
    if (family$family == "gaussian") {
    }
    if (family$family == "binomial") {
    }
    pred <- numeric()
    fit <- vector("list", length = 0)
    class(fit) <- c("SL.template")
    out <- list(pred = pred, fit = fit)
    return(out)
}
<environment: namespace:SuperLearner>
```