# Homework 3

*ST 697 | Fall 2017*

*Due: Thu Oct 19*

**Problem 3.1: Optimal Tuning Parameters**

In cross-validation, we discussed two ways to select the optimal tuning parameters. The first way is to use the value(s) that minimized the cross-validation error. The "one-standard error" rule uses the value(s) corresponding to the least complex model whose cv error is within one standard error of the best model. The goal of this assignment is to compare these two rules.

Use the simulated data from `mlbench.friedman1()` in the `mlbench` R package and fit elastic net models. The tuning parameter $\lambda$ (corresponding to the penalty on the coefficient magnitude) is the one we will focus one. Generate training data, use k-fold cross-validation to get $\lambda_{\text{min}}$ and $\lambda_{\text{1SE}}$, generate test data, make predictions for the test data, and compare performance of the two rules using mean squared error.

~~Check for any differences due to:~~ Choose reasonable values for:

- Choice of $K$ (number of folds)
- Number of training and test observations
- $\alpha$ (elasticnet tuning parameter)

This pseudo code will get you started:

```
library(mlbench)
library(glmnet)

#-- Settings
n.train =        # number of training obs
n.test =         # number of test obs
K =              # number of CV folds
alpha =          # glmnet tuning alpha (1 = lasso, 0 = ridge)
M =              # number of simulations

#-- Data Generating Function
getData <- function(n) mlbench.friedman1(n, sd=2) # data generating function

#-- Simulations

for(m in 1:M){

# 1. Generate Training Data
# 2. Build Training Model using cross-validation, e.g., cv.glmnet()
# 3. get lambda that minimizes cv error and 1 SE rule
# 4. Generate Test Data
# 5. Predict y values for test data (for each model)
# 6. Evaluate predictions

}

#-- Compare
# compare performance of the approaches
```

### Problem 3.2: Logistic Regression

Create a function that implements Logistic Regression.

- The function should have inputs: a predictor matrix $X$ (not including intercept), a vector $Y$, and some arguments related to convergence criteria.
- The function should have outputs: $\beta$ (and intercept).
- Test with an existing logistic regression function (e.g. `glm()`) to ensure you get the same coefficient values.
- Use any algorithm (newton, gradient, coordinate, etc.)
  - Test the speed of your algorithm on a data set with $n = 50{,}000$ and $p = 100$.

### Problem 3.3: Elastic Net

Create a function that implements the elastic net model (ESL 3.8.6 describes lasso algorithm). Compare your results to `glmnet()` from the `glmnet` package. Take note how `glmnet()` specifies the loss function and penalty. (Also, see the notes from class)

Papers that give more detail:

- Regularization Paths for Generalized Linear Models via Coordinate Descent
- Pathwise coordinate optimization