BIOST 546
WINTER QUARTER 2020

Homework # 4
Due Via Online Submission to Canvas: Fri, March 6 at 12 PM (Noon)

Instructions:

You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed.

Please remember — the easier you make it for the TA to find your answer, the easier it will be for him to give you credit for the problem!

On this assignment, some of the problems involve random number generation. Be sure to set a random seed (using the command `set.seed()`) before you begin.

1. In this exercise, you will generate simulated data, and will use this data to perform the lasso. Make sure you set a random seed before you begin.

   (a) Use the `rnorm()` function to generate a predictor X of length $n = 30$, and a noise vector ϵ of length $n = 30$.

   ```
   set.seed(1234)
   x <- rnorm(30)
   epsilon <- rnorm(30)
   ```

   (b) Generate a response vector Y of length $n = 30$ according to the model

   $$Y = 3 - 2X + 3 * X^2 + \epsilon.$$

   ```
   y <- 3-2*x+3*x^2+epsilon
   ```

   (c) Fit a lasso model to the data, using $X, X^2, \ldots, X^7$ in order to predict Y.

      i. Make a plot that displays the value of each coefficient, as a function of λ. You can display λ on the x-axis and "Coefficient Value" on the y-axis. Your plot should look something like the right-hand-side of

Figure 6.13 of the textbook, but with λ on the x-axis. Make sure that you display each coefficient in a different color, and use a caption or legend that clearly indicates which coefficient is which.

```
library(glmnet)
fit.lasso <- glmnet(X_feature, y, alpha=1,lambda.min.ratio = 0.000001)
plot(fit.lasso,xvar='lambda',label=TRUE, xlab=TeX('Log $\\lambda$'))
```

We included an example figure below; note that the default choices of λ in `glmnet` is likely to give you only two lines instead of all 7 lines. Try specifying a smaller sequence of λs by changing `lambda.min.ratio`

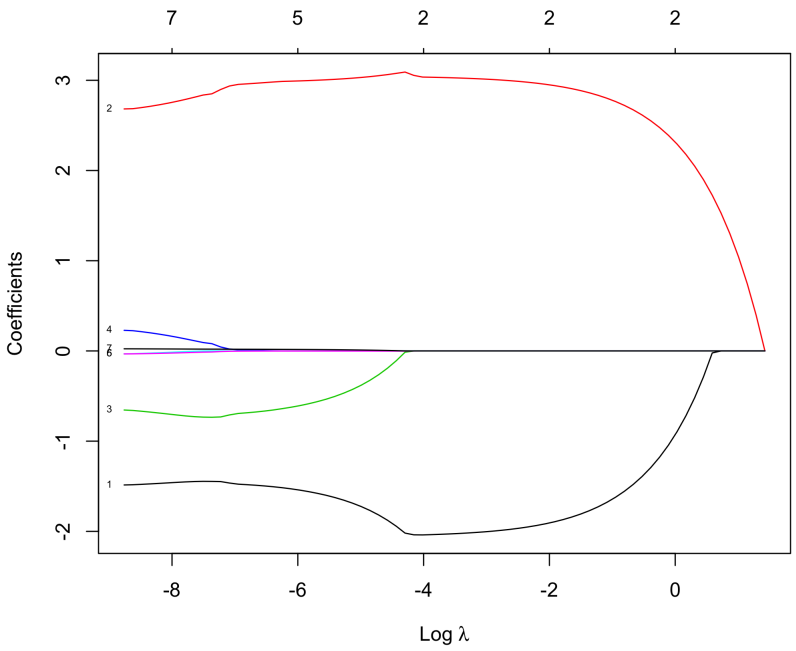

Figure 1: Standardized lasso regression coefficients as a function of log λ

ii. Use cross-validation to select the tuning parameter. What tuning parameter value do you choose? Make a plot to justify your choice. Your plot could display "Estimated Test Error" on the y-axis and λ on the x-axis (or it could display other quantities of your choice).

We used 5-fold cross-validation to select the tuning parameter. The λ that minimizes the CV MSE is 0.033

```
set.seed(1234)
fit.lasso.cv <- cv.glmnet(X_feature, y, type.measure="mse", alpha=1,
                          family="gaussian",nfolds=5)
plot(fit.lasso.cv)
```

iii. Fit a lasso model to all n observations, using the tuning parameter value selected in the previous sub-problem. Write out the fitted

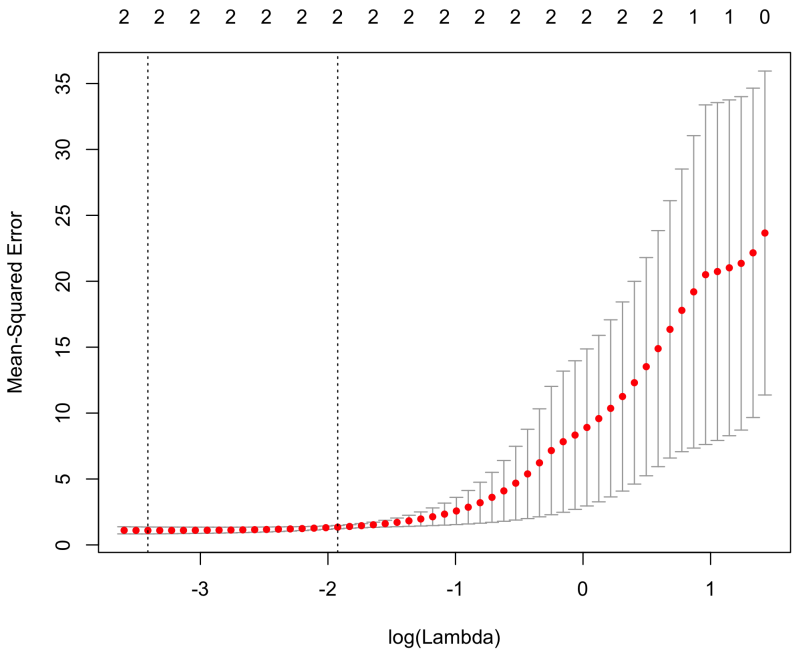

Figure 2: 5-fold cross-validated MSE for lasso regression as a function of $\log \lambda$

model. Comment on the fitted model.

```
fit.lasso.best.lamdba <- glmnet(X_feature, y, alpha=1,
lambda = fit.lasso.cv$lambda.min)
coef(fit.lasso.best.lamdba)
```

The fitted lasso model is $\hat{y} = 2.42 - 2.02x + 3.02x^2$ — really close to our true model. The lasso managed to zero out all the "spurious" relationships from x^3 to x^7!

(d) Now generate 1000 new observations generated according to 1(a) and 1(b). Apply the final fitted model from 1c(iii) to these new observations. What is the mean squared error?

```
set.seed(1234)
x_test <- rnorm(1000)
epsilon_test <- rnorm(1000)
y_test <- 3-2*x_test+3*x_test^2+epsilon_test
y_hat_lasso <- predict(fit.lasso.best.lamdba,
newx = poly(x_test,7,raw = TRUE))
mse_lasso <- mean((y_hat_lasso-y_test)^2)
```

The mean squared error for the lasso model on the new observations is 1.29

2. In this exercise, you will apply ridge regression to the data that you generated in 1(a), 1(b), and 1(d). (Make sure you use the same random seed!)

(a) Fit a ridge regression model to the data from 1(a) and 1(b), using $X, X^2, \ldots, X^7$ in order to predict Y.

```
fit.ridge <- glmnet(X_feature, y, alpha=0, lambda.min.ratio = 0.00001,
 nlambda=100)
```

i. Make a plot that displays the value of each coefficient, as a function of λ. You can display λ on the x-axis and "Coefficient Value" on the y-axis. Your plot should look something like the right-hand-side of Figure 6.12 of the textbook. Make sure that you display each coefficient in a different color, and use a caption or legend that clearly indicates which coefficient is which.

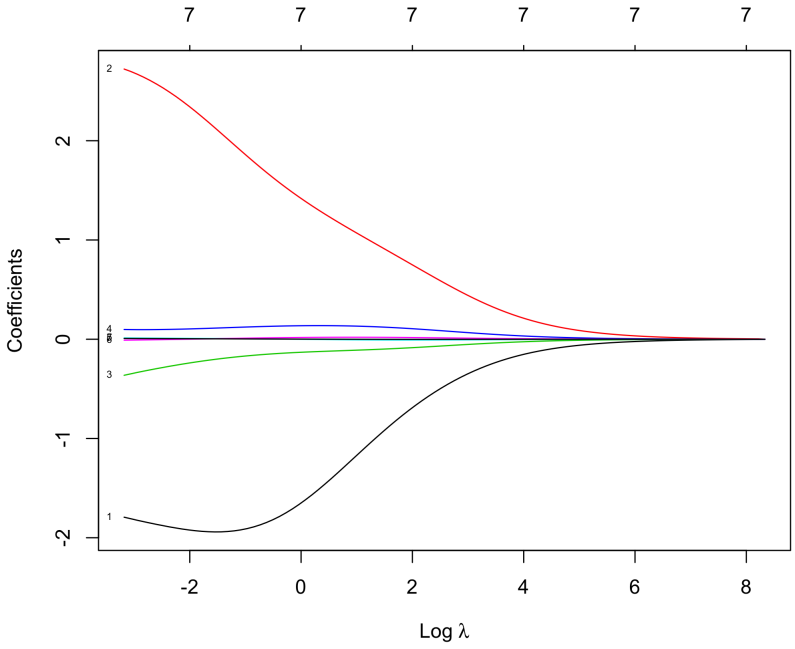

Figure 3: Standardized ridge regression coefficients as a function of $\log \lambda$

```
plot(fit.ridge,xvar='lambda',label=TRUE, xlab=TeX('Log $\\lambda$'))
```

ii. Use cross-validation to select the tuning parameter. What tuning parameter value do you choose? Make a plot to justify your choice. Your plot could display "Estimated Test Error" on the y-axis and λ on the x-axis (or it could display other quantities of your choice).
We used 5-fold cross-validation to select the tuning parameter. The λ that minimizes the CV MSE is 0.457

```
set.seed(1234)
fit.ridge.cv <- cv.glmnet(X_feature, y, type.measure="mse", alpha=0,
                          family="gaussian", nfolds = 5)
plot(fit.ridge.cv)
```

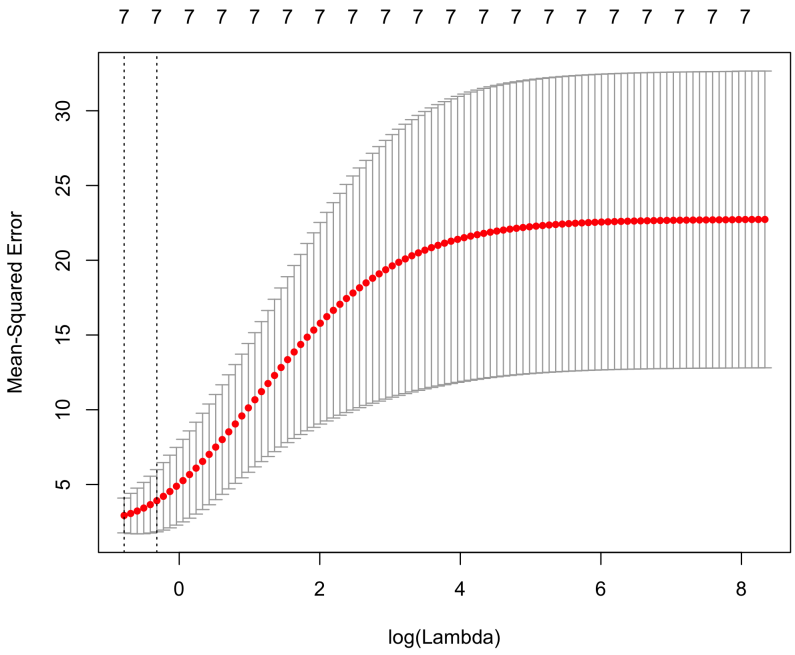

Figure 4: 5-fold cross-validated MSE for ridge regression as a function of $\log \lambda$

iii. Fit a ridge regression model to all n observations, using the tuning parameter value selected in the previous sub-problem. Write out the fitted model. Comment on the fitted model.

```
fit.ridge.best.lamdba <- glmnet(X_feature, y, alpha=0,
lambda = fit.ridge.cv$lambda.min)
coef(fit.ridge.best.lamdba)
```

The fitted ridge model is $\hat{y} = 3.03 - 1.87x + 1.75x^2 - 0.16x^3 + 0.13x^4 + 0.004x^5 + 0.012x^6 + 0.004x^7$. While the coefficients for x^3 to x^7 are small; ridge did not zero out the coefficients for these features, as one would expect.

(b) Apply the final fitted model from 2a(iii) to the observations generated in 1(d). What is your mean squared error?

```
y_hat_ridge <- predict(fit.ridge.best.lamdba,
newx = poly(x_test,7,raw = TRUE))
mse_ridge <- mean((y_hat_ridge-y_test)^2)
```

The mean squared error for the ridge model on the new observations is 1.29

(c) Now fit a least squares model to the data generated in 1(a) and 1(b), using $X, X^2, \ldots, X^7$ in order to predict Y. Then apply this fitted model to the 1000 new observations generated in 1(d). What mean squared error do you get?

```
fit.lm <- lm(y~., data = data.frame(X_feature, y))
coef(fit.lm)
y_hat_lm <- predict(fit.lm,
newdata = data.frame(poly(x_test,7,raw = TRUE)))
mse_lm <- mean((y_hat_lm-y_test)^2)
```

The fitted least squares model is $\hat{y} = 2.56 - 1.96x + 2.19x^2 + 0.37x^3 + 0.67x^4 - 0.53x^5 - 0.098x^6 + 0.082x^7$. The mean squared error for the least squares model is 136.23

(d) Given your answers to 1(d) and 2(b) and 2(c), which is a better choice on this data — ridge regression or the lasso or least squares? **The lasso model gives the smallest test set MSE and seems to be the best choice on this data, followed by ridge, and then least squares. This is not surprising since only 2 out of the 7 features are in the true data generating model.**

3. In this problem, we will simulate some data, and we'll compare the results that we get using least squares linear regression, and using a regression tree. Let $n = 100$ and $p = 2$.

(a) Generate $n = 100$ observations according to the linear model,

$$Y = 1 + 2X_1 + 3X_2 + \epsilon.$$

You can generate X_1, X_2, and ϵ independently from a $N(0, 1)$ distribution.

```
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100)
epsilon <- rnorm(100)
y = 1+2*x1+3*x2+epsilon
```

(b) Make a plot of the data. One axis of your plot should represent X_1, one axis should represent X_2, and the color of each point should represent the value of Y. You can use a command like this one:
`plot(x1, x2, col=rainbow(200)[rank(y)], pch=15)`.

(c) Do you expect least squares linear regression or a regression tree to perform better on this data, in terms of test error? Explain your answer.

Since the data were generated using a linear model (instead of a piece-wise constant function), we would expect the least squares linear regression to perform better on this data.

(d) Fit a least squares linear model to the data, and fit a regression tree to the data. (Be sure to prune your tree, if appropriate!) Report the fitted model for the former, and display the tree for the latter. You should of course make sure that the nodes in your regression tree are labeled appropriately.

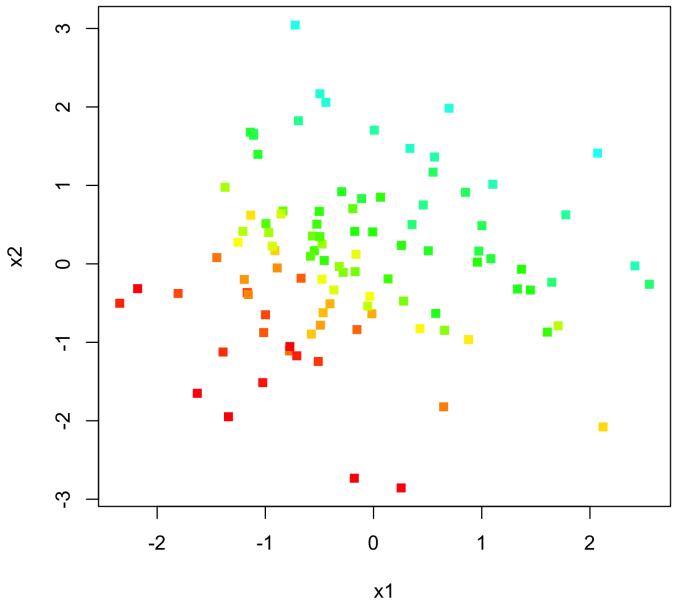

Figure 5: Generated training data for Q3

```
library(tree)
lm_q3 <- lm(y~., data = data.frame(x1,x2, y))
set.seed(1234)
tree_q3 <- tree(y~., data = data.frame(x1,x2,y))
cv_tree_q3 <- cv.tree(tree_q3)
plot(cv_tree_q3$size, cv_tree_q3$dev / length(y), type = "b",
     xlab = "Tree Size", ylab = "CV-MSE")
best_size <- cv_tree_q3$size[which.min(cv_tree_q3$dev)]
prune_tree_q3 <- prune.tree(tree_q3,best=best_size)
```

The fitted linear regression model is $\hat{y} = 1.16 + 2.08 \cdot x_1 + 3.09 \cdot x_2$. To pick the best tree model, we used the `cv.tree` function to estimate the cross-validated MSE of trees of different sizes. In this particular case, an unpruned (11 leaf nodes tree) tree gives the best prediction result. We plotted the fitted tree in Figure 6.

(e) Repeat the plot from (b), but this time display the partitions of feature space corresponding to the tree from (d). Furthermore, label each region with the predicted response value in this region. Your plot should look like Figure 8.2 in the textbook, except for two changes:

- the *predicted response value* in each region should also be displayed.
- the color of each observation should reflect the corresponding response value, as mentioned in (b).

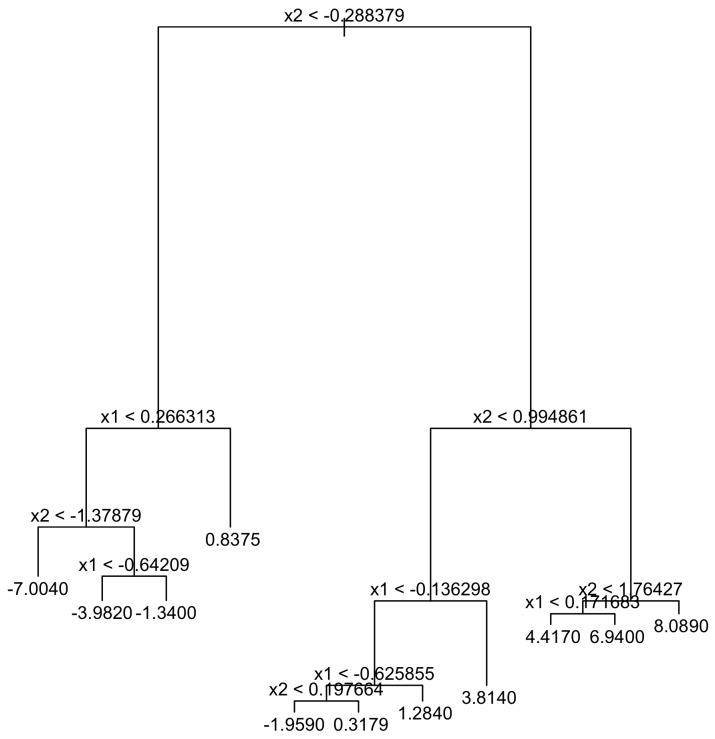

Figure 6: Fitted regression tree for Q3

```
partition.tree(prune_tree_q3,add = F)
points(x1, x2, col=rainbow(200)[rank(y)], pch=15)
```

(f) Generate 1000 test observations, and report the test error for both of the models that you fit in (d). Comment on your results.

```
set.seed(1234)
x1_test <- rnorm(1000)
x2_test <- rnorm(1000)
epsilon_test <- rnorm(1000)
X_test <- data.frame(x1_test,x2_test)
colnames(X_test) <- c('x1','x2')
y_test = 1+2*x1_test+3*x2_test+epsilon_test
tree_pred_q3 <- predict(prune_tree_q3,
newdata = X_test)
lm_pred_q3 <- predict(lm_q3,
newdata = X_test)
```

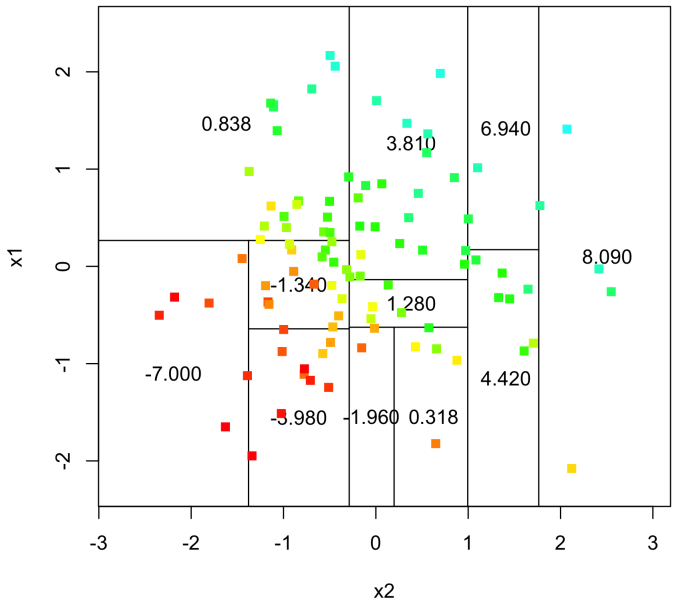

Figure 7: Paritions of feature space by the regression tree in Q3

```
mean((tree_pred_q3-y_test)^2)
mean((lm_pred_q3-y_test)^2)
```

The estimated MSEs for the linear model and regression tree are 1.06 and 3.78 respectively, which is expected based on the data generating mechanism.

4. Repeat the previous problem, but this time generate data as follows:

$$Y = 2 + 3I_{(X_1<0)} + 0.5I_{(X_1\geq 0, X_2<0.5)} - 2I_{(X_1\geq 0, X_2\geq 0.5)} + \epsilon.$$

Here, $I_{(A)}$ is an indicator variable that equals 1 if the event A holds, and equals 0 otherwise.

Since the data were generated under a piece-wise constant function, we would expect the regression tree to perform better on this data.

We plot simulated training data in Figure 8 and the resulted tree in Figure 9. This time `cv.tree` selected a node size of 3, which agrees with the true data generating mechanism. The feature space partitions can be found in Figure 10.

The MSE on test observations are 1.10 and 2.80 for the regression tree and linear model respectively.

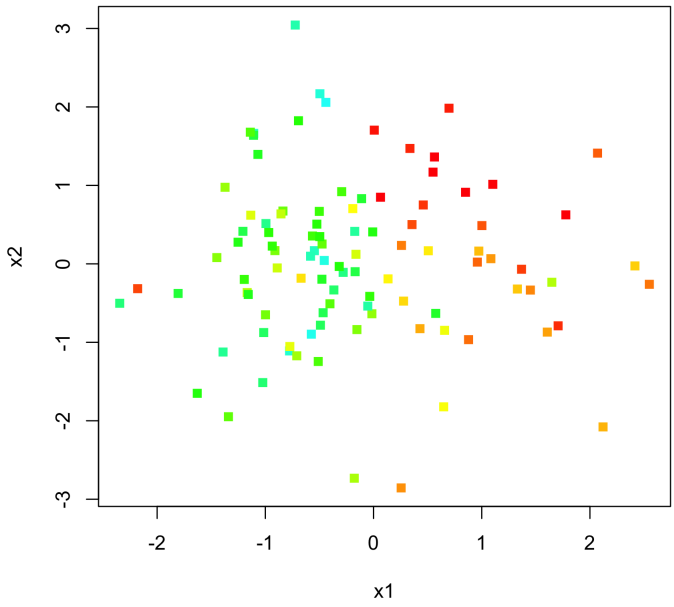

Figure 8: Generated training data for Q4

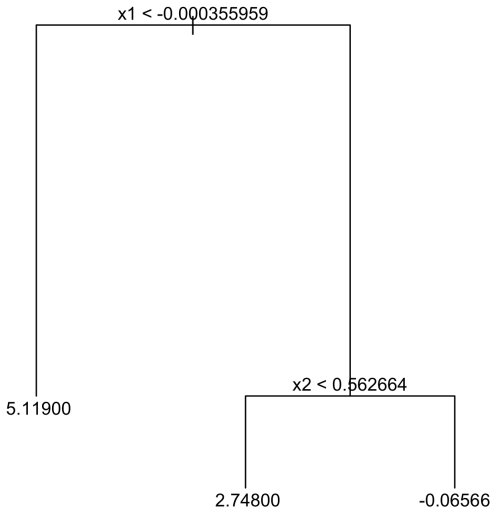

Figure 9: Fitted regression tree for Q4

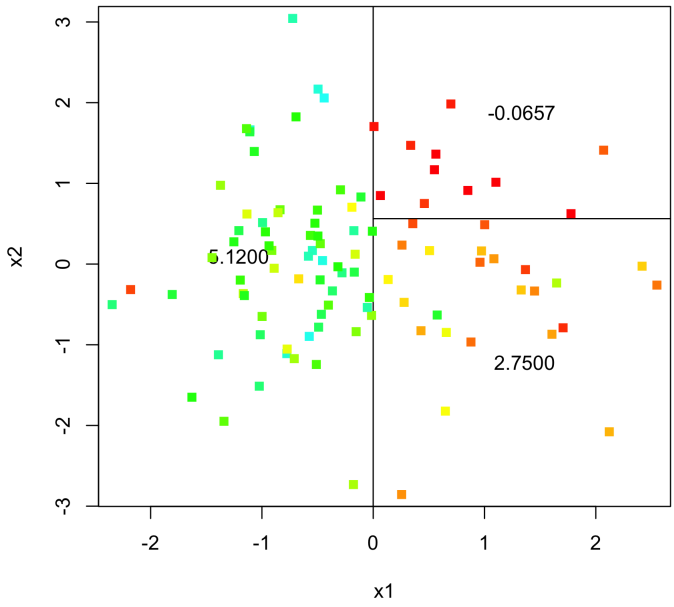

Figure 10: Paritions of feature space by the regression tree in Q4

```
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100)
eps <- rnorm(100)
y <- 2+3*(x1<0)+0.5*(x1>=0)*(x2<0.5)-2*(x1>=0)*(x2>=0.5)+eps
plot(x1, x2, col=rainbow(200)[rank(y)], pch=15)
lm_q4 <- lm(y~., data = data.frame(x1,x2, y))
set.seed(1234)
tree_q4 <- tree(y~., data = data.frame(x1,x2,y))
cv_tree_q4 <- cv.tree(tree_q4)
plot(cv_tree_q4$size, cv_tree_q4$dev / length(y), type = "b",
     xlab = "Tree Size", ylab = "CV-MSE")
best_size <- cv_tree_q4$size[which.min(cv_tree_q4$dev)]
prune_tree_q4 <- prune.tree(tree_q4,best=best_size)

#### q4 test set
set.seed(1234)
x1_test <- rnorm(1000)
x2_test <- rnorm(1000)
X_test <- data.frame(x1_test,x2_test)
colnames(X_test) <- c('x1','x2')
epsilon_test <- rnorm(1000)
y_test = 2+3*(x1_test<0)+0.5*(x1_test>=0)*(x2_test<0.5)-
```

```
2*(x1_test>=0)*(x2_test>=0.5)+epsilon_test
tree_pred_q4 <- predict(prune_tree_q4,newdata = X_test)
lm_pred_q4 <- predict(lm_q4,newdata = X_test)
mean((tree_pred_q4-y_test)^2)
mean((lm_pred_q4-y_test)^2)
```

5. **EXTRA CREDIT.** Let's consider doing least squares and ridge regression under a very simple setting, in which $p = 1$, and $\sum_{i=1}^n y_i = \sum_{i=1}^n x_i = 0$. We consider regression without an intercept. (It's usually a bad idea to do regression without an intercept, but if our feature and response each have mean zero, then it is okay to do this!)

   (a) The least squares solution is the value of $\beta \in \mathbb{R}$ that minimizes

   $$\sum_{i=1}^n (y_i - \beta x_i)^2.$$

   Write out an analytical (closed-form) expression for this least squares solution. Your answer should be a function of $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. *Hint: Calculus!!* **First-order optimality condition tells us that, at the minimizer, our objective function has derivative 0:**

   $$\frac{d}{d\beta} \sum_{i=1}^n (y_i - \beta x_i)^2 = 0 \implies \sum_{i=1}^n 2(y_i - \beta x_i) \cdot (-x_i) = 0$$

   $$\implies \sum_{i=1}^n x_i y_i - x_i^2 \beta = 0 \implies \hat{\beta}_{LS} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$$

   Since the original function is convex in β, this indeed will be the unique minimizer.

   (b) For a given value of λ, the ridge regression solution minimizes

   $$\sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda \beta^2.$$

   Write out an analytical (closed-form) expression for the ridge regression solution, in terms of $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ and λ.

   First-order optimality condition tells us that, at the minimizer, our objective function has derivative 0:

   $$\frac{d}{d\beta} \sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda \beta^2 = 0 \implies \sum_{i=1}^n 2(y_i - \beta x_i) \cdot (-x_i) + 2\lambda\beta = 0$$

   $$\implies \sum_{i=1}^n x_i y_i = \sum_{i=1}^n x_i^2 \beta + \lambda\beta \implies \hat{\beta}_R = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda}$$

   Since the original function is convex in β, this indeed will be the unique minimizer.

(c) Suppose that the true data-generating model is

$$Y = 3X + \epsilon,$$

where ϵ has mean zero, and X is fixed (non-random). What is the expectation of the least squares estimator from (a)? Is it biased or unbiased?

For (x_i, y_i) drawn from the data generating model , we have $\mathbb{E}[y_i] = \mathbb{E}[3x_i + \epsilon_i] = 3x_i$

$$\mathbb{E}[\hat{\beta}_{LS}] = \mathbb{E}[\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}] \overset{a.}{=} \frac{\mathbb{E}[\sum_{i=1}^n x_i y_i]}{\sum_{i=1}^n x_i^2} \overset{b.}{=} \frac{\sum_{i=1}^n x_i \mathbb{E}[y_i]}{\sum_{i=1}^n x_i^2}$$
$$\overset{c.}{=} \frac{\sum_{i=1}^n 3x_i^2}{\sum_{i=1}^n x_i^2} = 3$$

where $a.$ follows from the fact that $\sum_{i=1}^n \frac{1}{x_i^2}$ is just a constant; $b.$ follows from linearity of expectation, and $c.$ follows from the calculation of $\mathbb{E}[y_i]$ we did.

Since $\mathbb{E}[\hat{\beta}_{LS}] = 3 = \beta$, the least squares estimator is unbiased.

(d) Suppose again that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero, and X is fixed (non-random). What is the expectation of the ridge regression estimator from (b)? Is it biased or unbiased? Explain how the bias changes as a function of λ.

Following essentially the same calculations as above, we get

$$\mathbb{E}[\hat{\beta}_R] = \mathbb{E}[\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda}] = \frac{\mathbb{E}[\sum_{i=1}^n x_i y_i]}{\sum_{i=1}^n x_i^2 + \lambda} = \frac{\sum_{i=1}^n x_i \mathbb{E}[y_i]}{\sum_{i=1}^n x_i^2 + \lambda}$$
$$= 3 \cdot \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i^2 + \lambda}$$

.

Since $\mathbb{E}[\hat{\beta}_R] = 3 \cdot \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i^2 + \lambda} < 3$, the ridge regression estimator is (negatively) biased. The bias increases as λ increases.

(e) Suppose that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero and variance σ^2, and X is fixed (non-random), and also $\text{Cov}(\epsilon_i, \epsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the least squares estimator from (a)?

The data generating mechanism implies that $\mathbf{Var}(y_i) = \mathbf{Var}(3x_i + \epsilon) = \mathbf{Var}(\epsilon) = \sigma^2$ since we assumed x_i to be non-random. Moreover, we have $\mathbf{Var}(\sum_{i=1}^n x_i y_i) = \sum_{i=1}^n \mathbf{Var}(x_i y_i) + \sum_{i \neq j} \mathbf{Cov}(x_i y_i, x_j y_j) = \sum_{i=1}^n \mathbf{Var}(x_i y_i)$ since we assumed y_i and y_j are uncorrelated.

Now

$$\mathbf{Var}(\hat{\beta}_{LS}) = \mathbf{Var}(\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}) = \frac{\mathbf{Var}(\sum_{i=1}^n x_i y_i)}{(\sum_{i=1}^n x_i^2)^2}$$
$$= \frac{\sum_{i=1}^n \mathbf{Var}(x_i y_i)}{(\sum_{i=1}^n x_i^2)^2} = \frac{\sigma^2 \sum_{i=1}^n x_i^2}{(\sum_{i=1}^n x_i^2)^2} = \frac{\sigma^2}{\sum_{i=1}^n x_i^2}$$

(f) Suppose that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero and variance σ^2, and X is fixed (non-random), and also $\text{Cov}(\epsilon_i, \epsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the ridge estimator from (b)? How does the variance change as a function of λ?

Following the same steps in (d), we have

$$\textbf{Var}(\hat{\beta}_R) = \textbf{Var}(\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda}) = \frac{\textbf{Var}(\sum_{i=1}^n x_i y_i)}{(\sum_{i=1}^n x_i^2 + \lambda)^2}$$

$$= \frac{\sum_{i=1}^n \textbf{Var}(x_i y_i)}{(\sum_{i=1}^n x_i^2 + \lambda)^2} = \frac{\sigma^2 \sum_{i=1}^n x_i^2}{(\sum_{i=1}^n x_i^2 + \lambda)^2} < \frac{\sigma^2}{\sum_{i=1}^n x_i^2} = \textbf{Var}(\hat{\beta}_{LS})$$

We see that since $\lambda > 0$, we always have $\textbf{Var}(\hat{\beta}_R) < \textbf{Var}(\hat{\beta}_{LS})$ and the variance decreases as λ increases.

(g) In light of your answers to parts (d) and (f), argue that λ in ridge regression allows us to control model complexity by trading off bias for variance.

Based on our calculations above, we see that $\lambda > 0$ in the ridge regression introduces bias and reduces variance. In particular, a small λ corresponds to low bias and high variance (complex models), and a big λ corresponds to high bias and low variance (simple models). Therefore we will be able to implicitly encode our bias-variance trade-off by varying λ.

Hint: For this problem, you might want to brush up on some basic properties of means and variances! For instance, if $Cov(Z, W) = 0$, then $Var(Z + W) = Var(Z) + Var(W)$. And if a is a constant, then $Var(aW) = a^2 Var(W)$, and $Var(a + W) = Var(W)$.