STAT S4240 002, Homework 4

Brian Weinstein (bmw2148)

August 13, 2015

Problem 1: James 6.8.1

(a) The model obtained through *best subset selection* has the smallest training RSS for $k \leq p$ predictors. This is because best subset selection tests all possible subsets of size k and chooses the one that minimizes the training RSS.

There is a chance that by using *forward stepwise selection* or *backward stepwise selection* we'd end up with the same optimal subset of k predictors (i.e., the one we obtain via best subset selection), but since the k-sized subset is dependent on the covariates included in the prior steps in each of these methods, there's no guarantee that this will be the case.

(b) Without either valuating each model on a testing set or estimating test error, none among the three models is guaranteed to have the lowest test RSS. These methods only choose the subset that results in the lowest training RSS, which in no way indicates if the model will also result in a low test RSS.

(c) i. True. In a model identified by forward stepwise selection, the subset of predictors in a given step (with k predictors) is the base upon which the next predictor is added in the subsequent step (with $k+1$ predictors).

ii. True. In a model identified by backward stepwise selection, the subset of predictors in a given step (with k predictors) is itself a subset of the predictors used in the previous step (with $k+1$ predictors).

iii. False. The forward stepwise and backward stepwise models using k predictors are not guaranteed to use the same subset of predictors. As such, once an additional variable is added to the forward stepwise model (or equivalently, once a variable is removed from the backward stepwise model), there's no guarantee that the variables in the k-variable backward stepwise model are a subset of the $k+1$-variable forward stepwise model.

iv. False, with similar reasoning to Part (iii). The k predictors in a forward stepwise model are the predictors that, at each incremental step, kept the error as low as possible. Similarly, the $k+1$ predictors in a backward stepwise model are the predictors at each incremental step that kept the error as low as possible. Since each subset of predictors depends on those used in the previous step, there's no guarantee (although it may happen to be the case) that a set of predictors obtained via forward selection is a subset of those obtained via backward selection.

v. False. The set of k predictors chosen in best subset selection are those that keep the error in the k-variable model as low as possible. Independently, the set of $k+1$ predictors chosen in best subset are those that keep the error in the $k+1$-variable model as low as possible. There's no guarantee that k of the variables included in the $(k+1)$-variable model are those that minimize the error in the k-variable model.

Problem 2: James 6.8.3

Estimating the regression coefficients in a linear regression model via

$$\underset{\hat{\beta}}{\text{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

for a given value of s.

As we increase s from 0:

(a) The training RSS will (**iv.**) steadily decrease. As we make the constraint on $\sum_{j=1}^{p} |\beta_j|$ less strict, the limitation on the size of $\hat{\beta}$ disappears and we eventually get the standard "least residuals" solution. At $s = 0$ we start with only the intercept β_0 non-zero. As we increase s we allow for additional non-zero coefficients, eventually including all coefficients in the model. With all coefficients included, we minimize the training error (while potentially overfitting).

(b) The testing RSS will (**ii.**) decrease initially, and then eventually start increasing in a U shape. At $s = 0$ we start with a highly-biased model with only the intercept β_0 non-zero. As we increase s we allow for *some* of the $\beta_{j \neq 0}$ to take non-zero values (since the constraint is the same as that used in the lasso), which reduces the bias and increases the variance, for an overall drop in RSS. Once we increase s enough, *all* of the $\beta_{j \neq 0}$ will take non-zero values and we're left with a model with very low bias, but very high variance, increasing the RSS.

(c) The variance will (**iii.**) steadily increase. The reasoning here is similar to that used in Part (ii): At $s = 0$ we start with a very inflexible model, and as we increase s to allow more non-zero coefficients, the model flexibility, and thus our variance, increases.

(d) The (squared) bias will (**iv.**) steadily decrease. The reasoning here is similar to that used in Part (ii): At $s = 0$ we start with a highly-biased, very inflexible model with only one coefficient, β_0. As we increase s to allow more non-zero coefficients, the model flexibility, and thus our squared bias, decreases.

(e) The irreducible error will (**v.**) remain constant. The irreducible error is a function of the data we're given, not the model we choose, so our choice of s has no effect on the size of the irreducible error.

Problem 3: James 6.8.5

Problem 3

(a)

Problem 4: James 8.4.5

10 bootstrapped samples produces estimates of $P(\text{Class is Red}|X)$ that yield 0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75.

Based on the majority vote approach, we'd classify this X as Red, since 6 of the 10 estimates have probabilities greater than 0.5.

Based on the average probability approach, we'd classify this X as Green, since the average probability from our 10 samples is 0.45 (< 0.5).

Problem 5: Federalist Papers Classification with Trees

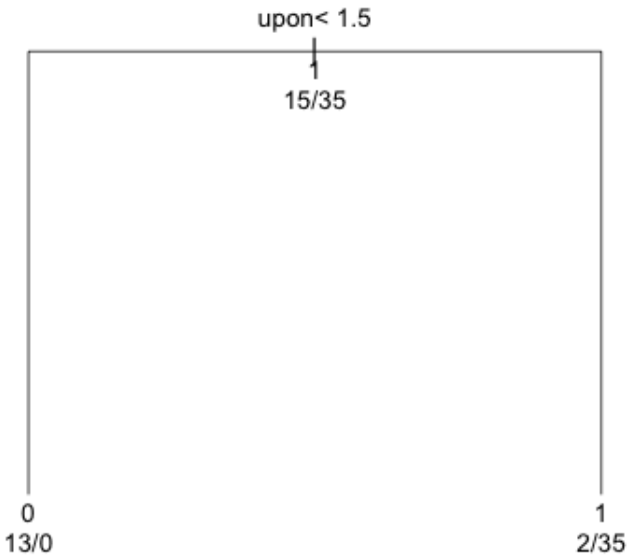
(a) Gini Impurity Coefficient Splits

i. Model Assessment

```
  accuracy falseNegRate falsePosRate
1 0.962963            0   0.09090909
```

- Accuracy: % of test papers that are classified correctly
- False Negative Rate: % of Hamilton papers incorrectly classified as Madison
- False Positive Rate: % of Madison papers incorrectly classified as Hamilton

ii. Tree Plot

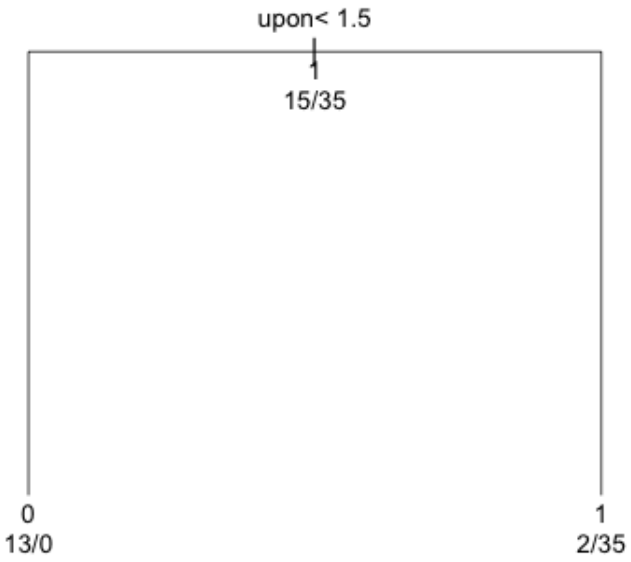


(b) Information Gain Splits

i. Model Assessment

```
  accuracy falseNegRate falsePosRate
1 0.962963            0   0.09090909
```

ii. Tree Plot



iii. The trees generated from Gini Impurity Coefficient splits and Information Gain splits ended up being identical here.

Problem 6: Federalist Papers Classification with Regularized Logistic Regression

(a) We should always center and scale our data whenever we us use a regularization method. Without scaling our data, the variables with larger values would have smaller coefficients (to indicate some level of importance), which would decrease the penalty for these predictors. Similarly, variables with smaller values would have much larger coefficients (to indicate the same level of importance), which would increase the penalty for these predictors.

In our dataset, since each of the word counts are on different scales (i.e., some words are much more common than others), performing unregularized logistic regression wouldn't penalize each of the variables in a comparable way.

(b) Ridge Regression

i. Model Assessment

```
   accuracy falseNegRate falsePosRate
1 0.5925926            0            1
```

ii. Important Words
The 10 words with the largest coefficients (in absolute value) are shown below.

```
       word        coef    absCoef
1  februari -0.01667849 0.01667849
2      upon  0.01572523 0.01572523
3    whilst -0.01428038 0.01428038
4    within -0.01367440 0.01367440
5     sever -0.01344406 0.01344406
6      1783 -0.01302052 0.01302052
7      form -0.01193143 0.01193143
8    member -0.01168190 0.01168190
9         5 -0.01153821 0.01153821
10 although -0.01127392 0.01127392
```

(c) Lasso

i. Model Assessment

```
   accuracy falseNegRate falsePosRate
1 0.8888889            0    0.2727273
```

ii. Important Words

The 10 words with the largest coefficients (in absolute value) are shown below.

```
       word       coef   absCoef
1    whilst -1.3557246 1.3557246
2  februari -1.1650795 1.1650795
3      upon  1.1493109 1.1493109
4      form -0.5714197 0.5714197
5  although -0.4741114 0.4741114
6    within -0.4227536 0.4227536
7     sever -0.3698089 0.3698089
8    lesser -0.3483888 0.3483888
9      anim -0.1694737 0.1694737
10 indirect -0.1641048 0.1641048
```

Both ridge and lasso include “februari”, “upon”, “whilst”, “within”, “sever”, “form”, and ”although” in the their 10 most important words.

“1783”, “member”, and “5” are included in the top 10 ridge words, but not in the top 10 lasso; and “lesser”, “anim”, and “indirect” are included in lasso, but not in ridge.

In general the ridge weights are smaller (in absolute value) than those from lasso.

Problem 7: Federalist Papers Classification with Feature Selection

(a) Problem 7

Problem 8: James 8.4.10

Using boosting to predict `Salary` in the `Hitters` data set.

(a) Removing the observations with no salary information, and log-transforming the salaries.

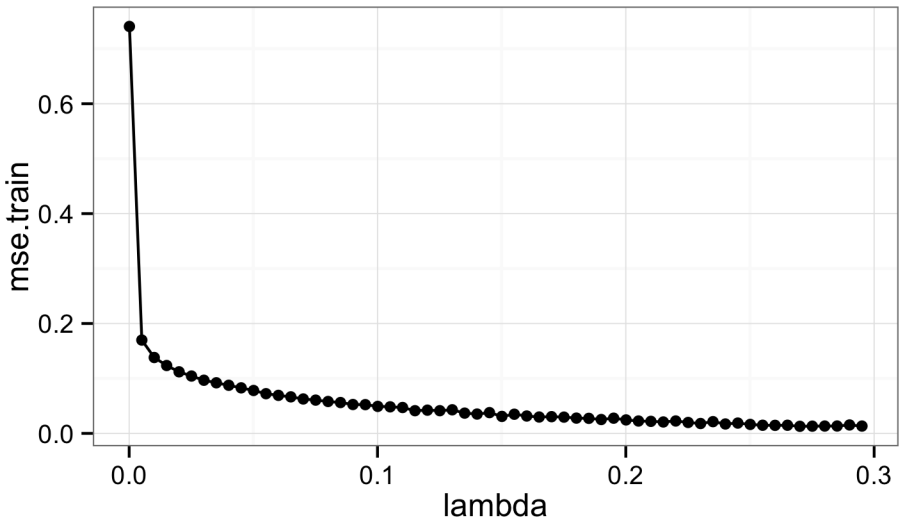
```
# remove observations with no salary info
Hitters <- Hitters[!is.na(Hitters$Salary), ]

# log transform the salaries
Hitters$Salary <- log(Hitters$Salary) #$
```

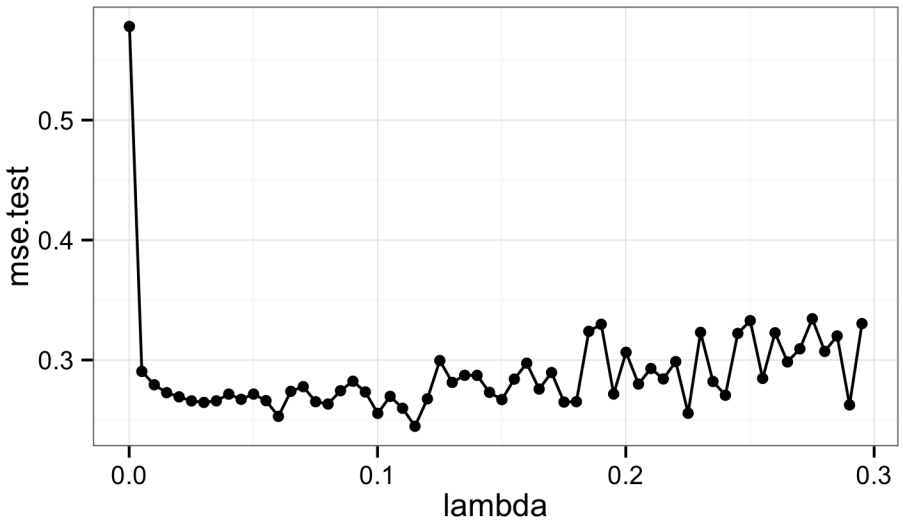
(b) Creating training and testing sets.

```
Hitters.train <- Hitters[1:200, ]
Hitters.test <- Hitters[-(1:200), ]
```

(c) Boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter $\lambda \in \{0.0001, 0.0051, 0.0101, 0.0151, \ldots, 0.2951\}$. Training MSE vs λ shown below.



(d) Boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter $\lambda \in \{0.0001, 0.0051, 0.0101, 0.0151, \ldots, 0.2951\}$. Test MSE vs λ shown below.



(e) Boosting generates a test MSE of 0.2447458 (at $\lambda = 0.1151$). I compared the boosting MSE to the errors obtained from OLS regression (from Chapter 3) and the lasso (from Chapter 6).

OLS regression generated a test MSE of 0.4918959:

```
model.ols <- lm(formula = Salary ~ ., data=Hitters.train)
pred.test.ols <- predict(model.ols, newdata=Hitters.test)
mse.test.ols <- mean((Hitters.test$Salary - pred.test.ols)^2) #$
```

And the lasso generates a test MSE of 0.4696547:

```
model.lasso <- cv.glmnet(x=model.matrix(Salary ~ ., Hitters.train)[, -1],
                         y=Hitters.train$Salary,
                         alpha=1, family="gaussian",
                         type.measure="mse")
pred.test.lasso <-  predict(model.lasso, newx=model.matrix(Salary ~ ., Hitters.test)[, -1],
                            s="lambda.min", type="response")
mse.test.lasso <- mean((Hitters.test$Salary - (as.data.frame(pred.test.lasso)$"1"))^2) #$
```

The boosting approach generates a significantly lower test MSE than OLS regression and the lasso.

(f) In the boosted model, `CAtBat` is by far the most important predictor. `PutOuts` is the second most important, but has less than half the relative influence of `CAtBat`. The relative influence statistics of all variables in the model are shown below.

```
                var     rel.inf
CAtBat       CAtBat 23.57153591
PutOuts     PutOuts  9.01152161
CHits         CHits  8.57868893
Walks         Walks  7.15658743
Years         Years  6.15196246
CRuns         CRuns  5.54809666
RBI             RBI  5.51043189
CRBI           CRBI  5.20052999
CHmRun       CHmRun  4.86688069
Assists     Assists  4.27511089
AtBat         AtBat  4.08888482
Hits           Hits  3.86999152
HmRun         HmRun  3.45383623
CWalks       CWalks  2.62813762
Errors       Errors  2.56909410
Runs           Runs  2.23166814
NewLeague NewLeague  0.66480707
Division   Division  0.52425937
League       League  0.09797467
```

(g) Bagging generates a test MSE of 0.2309305, which is slightly better than the test MSE obtained through the boosting and much better than the MSEs obtained through OLS regression and the lasso.

```
# train the model
model.bag <- randomForest(formula = Salary ~ .,
                          data=Hitters.train,
                          mtry=(ncol(Hitters.train) - 1), # consider *all* variables in
                                                         # each split (bagging)
                          ntree=1000,
                          importance=TRUE)

# use model to predict values for the testing data
pred.test.bag <- predict(model.bag,
                         newdata=Hitters.test[, -which(colnames(Hitters.test)=="Salary")])

# calculate the testing MSE
mse.test.bag <- mean((Hitters.test$Salary - pred.test.bag)^2) #$
```

Problem 9: James 10.7.1

Problem 9

This problem involves the K-means clustering algorithm.

(a) *Prove (10.12).*

(b) *On the basis of this identity, argue that the K-means clustering algorithm (Algorithm 10.1) decreases the objective (10.11) at each iteration.*

Todo list