Statistical Learning HW 4 - Ch 5

Christian Conroy

March 22, 2019

Book #9 a-c (3 points)

We will now consider the Boston housing data set, from the MASS library.

```
data("Boston")
```

(a) Based on this data set, provide an estimate for the population mean of medv.

```
muhat <- mean(Boston$medv)
muhat
```

```
## [1] 22.5
```

$\hat{\mu}$ is equal to 22.5

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result. Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

```
muhatse <- sd(Boston$medv)/sqrt(length(Boston$medv))
muhatse
```

```
## [1] 0.409
```

The standard error of $\hat{\mu}$ is equal to 0.409. (Make sure this shows up in RMD). This tells us how far the sample mean deviates from the actual population mean. Because the sample size is obviouly not that large, the SE is fairly high here.

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

```
set.seed(1)
meanFunc <- function(x,i){mean(x[i])}
bootMean <- boot(Boston$medv,meanFunc, 1000)
bootMean
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = meanFunc, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     22.5 0.00852       0.412
```

Using bootstrap, I end up with a standard rror of $\hat{\mu}$ equal to 0.412, which is higher than what I had originally calculated but pretty close. Given that the bootstrap method here is allowing us to emulate the process of obtaining new sample sets by repeatedly sampling observations from the original data set, this provides us

confidence in the standard error measurement. (Might be good to ask a question about this to Keegan and Jason)

Extra 38 (3 points)

Suppose we are given a training set with n observations and want to conduct k-fold cross-validation. Assume always that n = km where m is an integer. 100 =2m -> m = 50

a) Let k = 2. Explain carefully why there are 1/2 (n m) ways to partition the data into 2 folds.

In the case of k-fold cross validation, the first fold is treated as a validation set, and the method is fit on the remaining k-1 folds. In this case then, there is only one way to partition the data into two folds: One half of the data is validation and the other half is the train. In the equatin above, n represents the number of things to choose from and m represents how many of them we are choosing with no repetition and order does not matter. What the equation above signifies then is all the ways that we can apportion 100 into two groups of 50.

b) Let k = 3. Explain carefully why there are n!/3!m!m!m! ways to partition the data into 3 folds.

In this case, we will end up with one third of the data in validation and the other two-third split across two groups serving as the train. Like above, we are multiplying by 1/3 to signify that we are dividing into three groups. We are then using the n!/m!m!m! part to say that we have n observations of which to select (1/3*n) observations, which we do three times and hence the three factorials.

c) Guess a formula for the number of ways to partition the data into k folds for general k. Check if your formula gives the correct answer for k = n (leave-one-out c.v.)

1/k * (n!/k(m!^k))

Extra 41 (3 points)

Consider the built-in data set cars (see problem 1). We wish to predict the braking distance dist from speed. Use leave-one-out cross validation to find the best polynomial regression model. Repeat with 10-fold cross validation. Compare the two answers.

```
data("cars")
```

```
set.seed(1)
cv.error <- rep(0,5)
######### LOOCV
for (i in 1:5){
  glm.loocv <- glm(dist ~ poly(speed ,i),data=cars)
  cv.error[i]=cv.glm(cars, glm.loocv)$delta
}

cv.error
```

```
## [1] 246 243 247 250 280
```

```
#10 Fold
for (i in 1:5){
  glm.kf <- glm(dist ~ poly(speed ,i),data=cars)
  cv.error[i]=cv.glm(cars, glm.kf, K = 10)$delta
}

cv.error
```

```
## [1] 250 249 241 240 282
```

For LOOCV, we see a drop from the linear to the second order polynomial but then an increase after, so we'd conclude that the quadratic is likely the best model. For k-fold cross 10, by contrast, we end up with a drop through the fourth polynormial, leading us to believe that the fourth polynomial will be appropriate. Depending on how large the errors are, it may be appropriate however to select a lower polynomial from the K-fold conclusion, meaning that we'd likely reach a similar conclusion as to what we reached with LOOCV, namely that the second polynomial is appropriate.

Book 5 (5 points)

5. In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

```
data("Default")
```

(a) Fit a logistic regression model that uses income and balance to predict default.

```
logreg <- glm(default ~ income, data = Default, family=binomial)
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

```
table(Default$default)
```

```
##
##   No  Yes
## 9667  333
```

```
# We have 333 Yes and 9667 No, so it looks to be a classic class imbalance problem. Because of class im

Default_balanced <- tail(Default[order(Default$default),], -(9667-333))
table(Default_balanced$default)
```

```
##
##  No Yes
## 333 333
```

```
# While 80-20 is usually standard, given that we're now working with less data, move it to 70-30 here.
set.seed(1)
train <- sample(nrow(Default_balanced),(nrow(Default_balanced) * .70))

# Flaw obviously here is that I go from  10000 to 666 observations
```

ii. Fit a multiple logistic regression model using only the training observations.

```
lr.fit <- glm(default ~ income, data= Default_balanced, family=binomial, subset=train)
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

```
Default_test <- Default_balanced[-train,]
Default_test$pred <- predict(lr.fit, Default_test, type = "response")
Default_test$class_pred <- predict(lr.fit, Default_test, type = "response") > .5
```

```
summary(Default_test$pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.454   0.488   0.498   0.502   0.521   0.537
```

```
# Figure out why this may not be working!
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
sum(Default_test$class_pred == as.numeric(Default_test$default)) / nrow(Default_test)
```

```
## [1] 0.205
```

The test error accuracy is at 20.5%.

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
set.seed(17)
train <- sample(nrow(Default_balanced),(nrow(Default_balanced) * .80))

Default_test <- Default_balanced[-train,]
Default_test$pred <- predict(lr.fit, Default_test, type = "response")
Default_test$class_pred <- predict(lr.fit, Default_test, type = "response") > .5

sum(Default_test$class_pred == as.numeric(Default_test$default)) / nrow(Default_test)
```

```
## [1] 0.284
```

The test error is at 28.4%

```
set.seed(246)
train <- sample(nrow(Default_balanced),(nrow(Default_balanced) * .80))

Default_test <- Default_balanced[-train,]
Default_test$pred <- predict(lr.fit, Default_test, type = "response")
Default_test$class_pred <- predict(lr.fit, Default_test, type = "response") > .5

sum(Default_test$class_pred == as.numeric(Default_test$default)) / nrow(Default_test)
```

```
## [1] 0.224
```

The test error is at 22.4%

```
set.seed(349)
train <- sample(nrow(Default_balanced),(nrow(Default_balanced) * .80))

Default_test <- Default_balanced[-train,]
Default_test$pred <- predict(lr.fit, Default_test, type = "response")
Default_test$class_pred <- predict(lr.fit, Default_test, type = "response") > .5

sum(Default_test$class_pred == as.numeric(Default_test$default)) / nrow(Default_test)
```

```
## [1] 0.291
```

The test error is at 29.1%

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set

approach. Comment on whether or not including a dummy

```
lr.fit2 <- glm(default ~ income + balance + student, data= Default_balanced, family=binomial, subset=tra
```

```
Default_test <- Default_balanced[-train,]
Default_test$pred <- predict(lr.fit2, Default_test, type = "response")
Default_test$class_pred <- predict(lr.fit2, Default_test, type = "response") > .5
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
sum(Default_test$class_pred == as.numeric(Default_test$default)) / nrow(Default_test)
```

```
## [1] 0.0896
```

The test error is at approximatly 9.0%

Extra 39 (5 Points)

In this problem, we use the Advertising data. We want to predict Sales from TV, Radio and Newspaper, using multiple regression (no interaction terms, no polynomial terms). There are therefore three models with exactly one predictor, three with exactly two predictors, and one with all three predictors.

```
Advertising <- read.csv('Advertising.csv', header = TRUE, stringsAsFactors = FALSE)
head(Advertising)
```

```
##   X    TV Radio Newspaper Sales
## 1 1 230.1  37.8      69.2  22.1
## 2 2  44.5  39.3      45.1  10.4
## 3 3  17.2  45.9      69.3   9.3
## 4 4 151.5  41.3      58.5  18.5
## 5 5 180.8  10.8      58.4  12.9
## 6 6   8.7  48.9      75.0   7.2
```

a) Make all seven models. Do not show the summaries.

```
reg1 <- lm(Sales ~ TV, data= Advertising)
reg2 <- lm(Sales ~ Radio, data= Advertising)
reg3 <- lm(Sales ~ Newspaper, data= Advertising)
reg4 <- lm(Sales ~ TV + Newspaper, data= Advertising)
reg5 <- lm(Sales ~ TV + Radio, data= Advertising)
reg6 <- lm(Sales ~ Radio + Newspaper, data= Advertising)
reg7 <- lm(Sales ~ TV + Radio + Newspaper, data= Advertising)
```

b) There are six ways to nest these models from smallest (only one predictor) to largest (all three predictors). Carry out ANOVA comparisons for all six ways.

```
anova(reg1, reg2, reg3, reg4, reg5, reg6, reg7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ TV
## Model 2: Sales ~ Radio
## Model 3: Sales ~ Newspaper
## Model 4: Sales ~ TV + Newspaper
## Model 5: Sales ~ TV + Radio
## Model 6: Sales ~ Radio + Newspaper
```

```
## Model 7: Sales ~ TV + Radio + Newspaper
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1    198 2103
## 2    198 3618  0     -1516
## 3    198 5135  0     -1516
## 4    197 1919  1      3216 1132 <2e-16 ***
## 5    197  557  0      1362
## 6    197 3615  0     -3058
## 7    196  557  1      3058 1076 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

c) Summarize what you see. Is there a predictor that typically does not improve a model significantly if it is added? What are the models that are always improved significantly if another predictor is added? Is there anything that these models have in common?

There is statistical significance for the model with TV and Newspaper as predictors and the model with all three as predictors. It does not look like newspaper typically does not improve a model significantly if it is added. In fact, the RSS is the same for the model with TV and Radio and the model with all three predictors. The model with TV improves significantly each time another predictor is added.

Extra 40 (5 points)

In this problem, we use the Advertising data. We want to predict Sales from TV, Radio and Newspaper, using multiple regression with all three predictors plus up to one interaction term of these three predictors, e.g. TV * Radio or Radio * Newspaper. Should such an interaction term be included? Which one? Try to answer this question by estimating the residual standard error using 10-fold cross validation for all four possible models.

```
# TV
set.seed(1)
cv.error <- rep(0,2)
for (i in 1:2){
  lm.kf <- glm(Sales ~ poly(TV, i) + Radio + Newspaper, data= Advertising)
  cv.error[i]=cv.glm(Advertising, lm.kf, K = 10)$delta
}
```

```
## Warning in cv.error[i] <- cv.glm(Advertising, lm.kf, K = 10)$delta: number
## of items to replace is not a multiple of replacement length

## Warning in cv.error[i] <- cv.glm(Advertising, lm.kf, K = 10)$delta: number
## of items to replace is not a multiple of replacement length
```

```
cv.error
```

```
## [1] 2.93 2.43
```

```
# Radio
set.seed(1)
cv.error <- rep(0,2)
for (i in 1:2){
  lm.kf <- glm(Sales ~ TV + poly(Radio, i) + Newspaper, data= Advertising)
  cv.error[i]=cv.glm(Advertising, lm.kf, K = 10)$delta
}
```

```
## Warning in cv.error[i] <- cv.glm(Advertising, lm.kf, K = 10)$delta: number
```

```
## of items to replace is not a multiple of replacement length

## Warning in cv.error[i] <- cv.glm(Advertising, lm.kf, K = 10)$delta: number
## of items to replace is not a multiple of replacement length
```

```
cv.error
```

```
## [1] 2.93 2.98
```

```
# Newspaper
set.seed(1)
cv.error <- rep(0,2)
for (i in 1:2){
  lm.kf <- glm(Sales ~ TV + Radio + poly(Newspaper, i),data= Advertising)
  cv.error[i]=cv.glm(Advertising, lm.kf, K = 10)$delta
}
```

```
## Warning in cv.error[i] <- cv.glm(Advertising, lm.kf, K = 10)$delta: number
## of items to replace is not a multiple of replacement length

## Warning in cv.error[i] <- cv.glm(Advertising, lm.kf, K = 10)$delta: number
## of items to replace is not a multiple of replacement length
```

```
cv.error
```

```
## [1] 2.93 3.07
```

Including an interaction term for TV would be appropriate as its inclusion decreases the residual standard error while the residual standard error increases when including an interaction term for the other two predictors respectively.