HW4

Norman Hong

March 17, 2019

9

Consider the Boston housing data set, from the MASS library.

(a)

Based on this data set, provide an estimate for the population mean of medv. Call this estimate mu_hat.

```
# Boston data set has 506 observations, so we create a resample of same size.
set.seed(314)
mean.bootstrap <- function(data=Boston, index=nrow(data)){
  mean(data$medv[index]) # Same index can be used more than once.
}
# calculated mean of 1 resample.
mu_hat <- mean.bootstrap(Boston, sample(nrow(Boston), nrow(Boston), replace=TRUE))
cat('mean of a single resample',mu_hat)
```

```
## mean of a single resample 22.21957
```

```
cat('\n')
```

```
cat('mean of original sample', mean(Boston$medv))
```

```
## mean of original sample 22.53281
```

(b)

Provide an estimate of the standard error of mu_hat. Interpret this result.
(Calculate the standard deviation of the sample mean, which is mu_hat)

The standard deviation of the sample mean is about .4. This describes the spread of the sample mean random variable.

```
set.seed(314)
sd.bootstrap <- function(data, index){
  sd(data$medv[index])
}
sd.sample <- sd.bootstrap(Boston, sample(506, 506, replace=TRUE))
cat('standard error of the sample mean using a single resample', sd.sample/sqrt(nrow(Boston)))
```

```
## standard error of the sample mean using a single resample 0.395683
```

```
cat('\n')
```

```
cat('standard error of the sample mean using original sample', sd(Boston$medv)/sqrt(nrow(Boston)))
```

```
## standard error of the sample mean using original sample 0.4088611
```

(c)

Now estimate the standard error of mu_hat using the bootstrap. How does this compare to your answer from (b)?

The bootstrap estimate differs from the answer in b by a small margin. The boostrap estimate is .403 and the estiamte computed in (b) is .395.

```
set.seed(314)
boot(Boston, mean.bootstrap, 10000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston, statistic = mean.bootstrap, R = 10000)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 22.53281 -0.007551996   0.4034125
```

Extra 38

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.

##(a) Let $k = 2$. Explain carefully why there are $1/2 \cdot \binom{n}{m}$ ways to partition the data into 2 folds.

2 folds mean that when $k = 2$ we want to create a pair of 2 different group where the sum of the length of the 2 groups is n out of the n integers. So, k determines the number of groups to create. m determines the number of elements that belong to each group. There are $(n) \cdot (n-1) \cdot (n-2) \cdot ... \cdot 1$ ways to put n items into n spots. Because we're trying to group these n items into 2 groups, the order does not matter within the 2 groups. Therefore, we divide by $m! \cdot m!$. Since these 2 groups represent a pair, the order of these 2 groups does not matter. For example, $group1, group2 = group2, group1$. This means that we have to divide by $k = 2!$. Using algebra, this can be written into the nice form $1/2 \cdot \binom{n}{m}$.

(b)

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

3 folds mean that when $k = 3$ we want to create a group of 3 different group where the sum of the length of the 3 groups is n out of the n integers. So, k determines the number of groups to create. m determines the number of elements that belong to each group. There are $(n) \cdot (n-1) \cdot (n-2) \cdot ... \cdot 1$ ways to put n items into n spots. Because we're trying to group these n items into 3 groups, the order does not matter within the 3 groups. Therefore, we divide by $m! \cdot m! \cdot m!$. Since these 3 groups represent a single group, the order of these 3 groups does not matter. For example, $group1, group2, group3 = group2, group1, group3$. This means that we have to divide by $k = 3!$. Using algebra, this can be written into the nice form $n!/(3!m!m!m!)$.

(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.).

Formula: $n!/(k!)(\prod_{i=1}^{k} m_i!)$. The formula gives the correct answer for leave-one-out c.v. In leave-one-out, $k = n$ and $m = 1$. Therefore, the equation will equal to 1 because there is only 1 way to create a group of n

different groups where the sum of the length of the n groups is n and each of the n different groups have 1 element.

Extra 41

Consider the build-in data set cars. 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.

The results show that the two metheds have very similar scores. Both methods indicate that the models with low MSE correspond to degrees of 1 to 10. These models correspond to the best fit. However, it is best to use the most simpliest model possible, so the model with linear variable is the best.

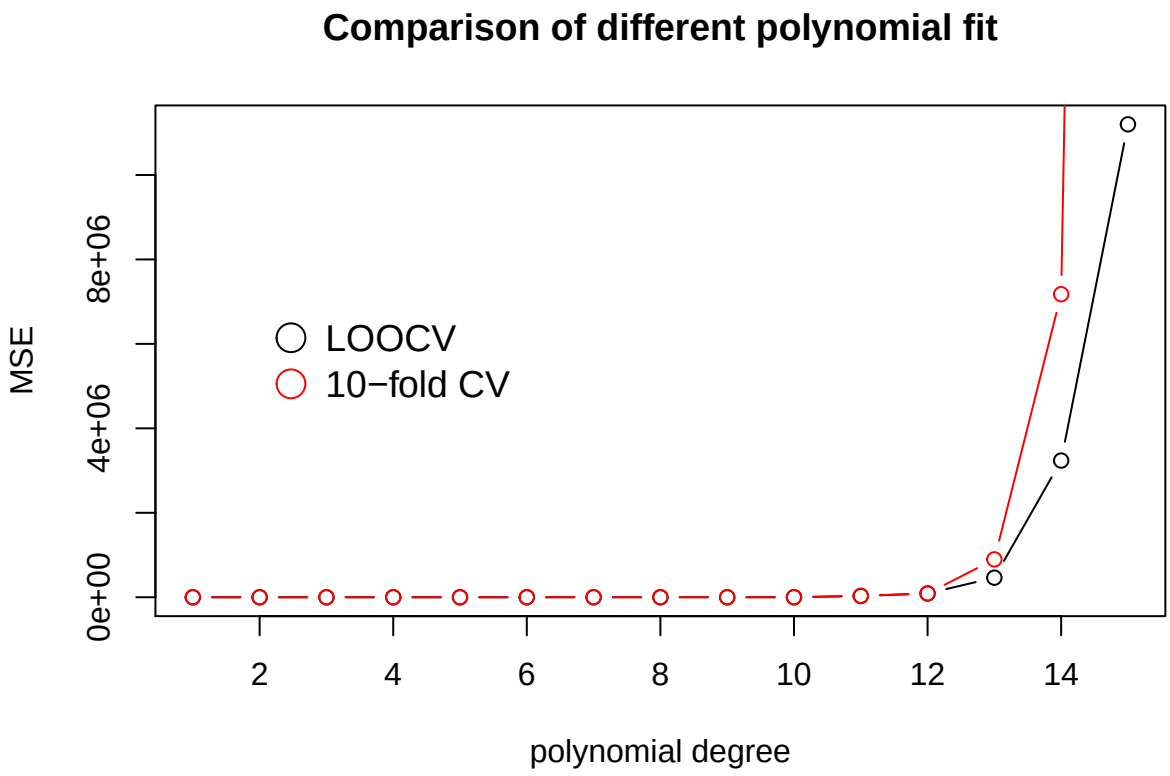
```
data("cars")
set.seed(314)
cv.score <- rep(NA, 15)

# Leave-1-out c.v.
for(k in 1:15){
  glm.fit <- glm(dist~poly(speed,k), data=cars) # using glm to do linear regression
  cv.err <- cv.glm(cars, glm.fit)
  cv.score[k] <- cv.err$delta[1] # delta[1] = non-bias corrected cv score.
}

plot(cv.score, type='b', main='Comparison of different polynomial fit',
     xlab='polynomial degree', ylab='MSE', col=1)

# 10-fold c.v.
cv.score <- rep(NA, 15)
for(k in 1:15){
  glm.fit <- glm(dist~poly(speed,k), data=cars) # using glm to do linear regression
  cv.err <- cv.glm(cars, glm.fit, K=10)
  cv.score[k] <- cv.err$delta[1] # delta[1] = non-bias corrected cv score.
}

lines(cv.score, type='b', col=2)
legend('left', legend=c('LOOCV', '10-fold CV'), col=c(1,2), pch=1,
  bty = "n",
  pt.cex = 2,
  cex = 1.2,
  text.col = "black",
  horiz = F ,
  inset = c(0.1, 0.1))
```



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.

(a)

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

```
fit.log <- glm(default~income+balance, data=Default, family='binomial')
summary(fit.log)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
```

```
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(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.

```
set.seed("314")
obs <- nrow(Default)
train <- sample(obs, obs/2, replace=FALSE) # 50-50 split in data.
```

ii)

Fit a multiple logistic regression model using only the training observations

```
Default.copy <- Default
Default.copy$default <- as.numeric(Default.copy$default == 'Yes') # 1 = 'Yes'
fit.log <- glm(default~income+balance, data=Default.copy, family="binomial", subset=train)
summary(fit.log)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default.copy, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3611  -0.1565  -0.0668  -0.0268   3.6297
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.093e+01  5.780e-01 -18.910  < 2e-16 ***
## income       2.081e-05  6.943e-06   2.998  0.00272 **
## balance      5.273e-03  2.999e-04  17.579  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1456.95  on 4999  degrees of freedom
## Residual deviance:  841.82  on 4997  degrees of freedom
## AIC: 847.82
##
## Number of Fisher Scoring iterations: 8
```

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 .5.

```
probs <- predict(fit.log, Default[-train,], type='response')
pred <- rep(0, length(probs))
pred[probs > .5] <- 1
```

iv)

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

```
mis <- pred != Default.copy$default[-train]
mean(mis)
```

```
## [1] 0.0254
```

(c)

Repeat the process in (b) 3 times, using 3 different splits of the observations into a training set and a validation set. Comment on the results obtained.

The validation set error changes depending on how the data was split into the 2 sets. There is wild fluctuations in the validation set error relative to the scale of the numbers.

```
set.seed(400)
for(i in 1:3){
  train <- sample(obs, obs/2, replace=FALSE)
  Default.copy <- Default
  Default.copy$default <- as.numeric(Default.copy$default == 'Yes') # 1 = 'Yes'
  fit.log <- glm(default~income+balance, data=Default.copy, family="binomial", subset=train)
  probs <- predict(fit.log, Default[-train,], type='response')
  pred <- rep(0, length(probs))
  pred[probs > .5] <- 1
  mis <- pred != Default.copy$default[-train]
  cat(mean(mis), '\n')
}
```

```
## 0.0284
## 0.0252
## 0.0274
```

(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 variable for student leads to a reduction in the test error rate.

I used the same seed to ensure that the split in the data was the same as the split in part c. The results indicate that the validation set error rate is the same. This implies that the model with the student dummy variable does not affect the model.

```
set.seed(400)
for(i in 1:3){
```

```
  train <- sample(obs, obs/2, replace=FALSE)
  Default.copy <- Default
  Default.copy$default <- as.numeric(Default.copy$default == 'Yes') # 1 = 'Yes'
  fit.log <- glm(default~., data=Default.copy, family="binomial", subset=train)
  probs <- predict(fit.log, Default[-train,], type='response')
  pred <- rep(0, length(probs))
  pred[probs > .5] <- 1
  mis <- pred != Default.copy$default[-train]
  cat(mean(mis), '\n')
}
```

```
## 0.0288
## 0.0252
## 0.0274
```

Extra 39

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). Make 3 models with exactly one predictor, 3 with exactly 2 predictors, and 1 with all 3 predictors.

(a)

Make all 7 models. Do not show the summaries.

```
fit.lm.1 <- lm(sales~TV, data=ads)
fit.lm.2 <- lm(sales~radio, data=ads)
fit.lm.3 <- lm(sales~newspaper, data=ads)
fit.lm.4 <- lm(sales~TV+radio, data=ads)
fit.lm.5 <- lm(sales~TV+newspaper, data=ads)
fit.lm.6 <- lm(sales~radio+newspaper, data=ads)
fit.lm.7 <- lm(sales~TV+radio+newspaper, data=ads)
```

##(b) There are 6 ways to nest these models from smallest (only 1 predictor) to largest (all 3 predictors). Carry out ANOVA comparisons for all 6 ways.

```
anova(fit.lm.1, fit.lm.4, fit.lm.7)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ TV
## Model 2: sales ~ TV + radio
## Model 3: sales ~ TV + radio + newspaper
##   Res.Df     RSS Df Sum of Sq        F Pr(>F)
## 1    198 2102.53
## 2    197  556.91  1   1545.62 544.0501 <2e-16 ***
## 3    196  556.83  1      0.09   0.0312 0.8599
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit.lm.1, fit.lm.5, fit.lm.7)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ TV
## Model 2: sales ~ TV + newspaper
```

```
## Model 3: sales ~ TV + radio + newspaper
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)
## 1    198 2102.53
## 2    197 1918.56  1    183.97  64.756  7.97e-14 ***
## 3    196  556.83  1   1361.74 479.325 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit.lm.2, fit.lm.4, fit.lm.7)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ radio
## Model 2: sales ~ TV + radio
## Model 3: sales ~ TV + radio + newspaper
##   Res.Df    RSS Df Sum of Sq         F Pr(>F)
## 1    198 3618.5
## 2    197  556.9  1   3061.57 1077.6574 <2e-16 ***
## 3    196  556.8  1      0.09    0.0312 0.8599
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit.lm.2, fit.lm.6, fit.lm.7)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ radio
## Model 2: sales ~ radio + newspaper
## Model 3: sales ~ TV + radio + newspaper
##   Res.Df    RSS Df Sum of Sq         F Pr(>F)
## 1    198 3618.5
## 2    197 3614.8  1      3.64    1.2828 0.2588
## 3    196  556.8  1   3058.01 1076.4058 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit.lm.3, fit.lm.5, fit.lm.7)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ newspaper
## Model 2: sales ~ TV + newspaper
## Model 3: sales ~ TV + radio + newspaper
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    198 5134.8
## 2    197 1918.6  1    3216.2 1132.10 < 2.2e-16 ***
## 3    196  556.8  1    1361.7  479.33 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit.lm.3, fit.lm.6, fit.lm.7)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ newspaper
## Model 2: sales ~ radio + newspaper
## Model 3: sales ~ TV + radio + newspaper
```

```
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    198 5134.8
## 2    197 3614.8  1      1520  535.02 < 2.2e-16 ***
## 3    196  556.8  1      3058 1076.41 < 2.2e-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?

Adding radio and TV variables always lead to a statistical significant increase in the fit of the model. Adding newspaper variable to the model does not always improve a model significantly. Adding newspaper improves the model when there is only the TV variable in the model. The results point to the idea that TV and radio are the two best predictor variables to use in the model.

Extra 40

In this problem, we use the Advertising data. We want to predict Sales from TV, Radio, and Newspaper, using multiple regression with all 3 predictors plus up to 1 interaction term of these 3 predictors ($TV \cdot radio$ or $Radio \cdot newspaper$ or $TV \cdot newspaper$). 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 4 possible models.

The model with the interaction term $TV \cdot Radio$ should be included because that model had the lowest cv MSE and cv RSE.

```
set.seed(341)
fit.lm.1 <- glm(sales~TV+radio+newspaper, data=ads)
fit.lm.2 <- glm(sales~TV+radio+newspaper+TV*newspaper, data=ads)
fit.lm.3 <- glm(sales~TV+radio+newspaper+radio*newspaper, data=ads)
fit.lm.4 <- glm(sales~TV+radio+newspaper+TV*radio,data=ads)

# 10-fold c.v. using MSE metric on training data
temp <- list(fit.lm.1, fit.lm.2, fit.lm.3, fit.lm.4)
for(i in temp){
  cv.err <- cv.glm(ads, i, K=10)
  cat(cv.err$delta[1], '\n') # delta[1] = non-bias corrected cv score.
}
```

```
## 2.909706
## 2.852686
## 3.006746
## 0.9784691
```

```
# 10-fold c.v. using RSE metric
cv.fit.lm.1 <- function (){
#Randomly shuffle the data
ads.copy <- ads[sample(nrow(ads)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(ads.copy)),breaks=10,labels=FALSE)
#Perform 10 fold cross validation
cv.score <- rep(NA, 10)
for(i in 1:10){
```

```
    #Segement your data by fold using the which() function
    testIndexes <- which(folds==i, arr.ind=TRUE)
    testData <- ads.copy[testIndexes,]
    trainData <- ads.copy[-testIndexes,]
    model <- glm(sales~TV+radio+newspaper, data=trainData)
    pred <- predict(model, testData, type='response')
    residuals <- (pred - testData$sales)
    res.sq <- residuals**2
    cv.score[i] <- sqrt(sum(res.sq)/model$df.residual)
}
cat(mean(cv.score))
}
cv.fit.lm.1()
```

```
## 0.5710154
```

```
cv.fit.lm.2 <- function (){
#Randomly shuffle the data
ads.copy <- ads[sample(nrow(ads)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(ads.copy)),breaks=10,labels=FALSE)
#Perform 10 fold cross validation
cv.score <- rep(NA, 10)
for(i in 1:10){
    #Segement your data by fold using the which() function
    testIndexes <- which(folds==i, arr.ind=TRUE)
    testData <- ads.copy[testIndexes,]
    trainData <- ads.copy[-testIndexes,]
    model <- glm(sales~TV+radio+newspaper+TV*newspaper, data=trainData)
    pred <- predict(model, testData, type='response')
    residuals <- (pred - testData$sales)
    res.sq <- residuals**2
    cv.score[i] <- sqrt(sum(res.sq)/model$df.residual)
}
cat(mean(cv.score))
}
cv.fit.lm.2()
```

```
## 0.5637144
```

```
cv.fit.lm.3 <- function (){
#Randomly shuffle the data
ads.copy <- ads[sample(nrow(ads)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(ads.copy)),breaks=10,labels=FALSE)
#Perform 10 fold cross validation
cv.score <- rep(NA, 10)
for(i in 1:10){
    #Segement your data by fold using the which() function
    testIndexes <- which(folds==i, arr.ind=TRUE)
    testData <- ads.copy[testIndexes,]
    trainData <- ads.copy[-testIndexes,]
    model <- glm(sales~TV+radio+newspaper+radio*newspaper, data=trainData)
    pred <- predict(model, testData, type='response')
    residuals <- (pred - testData$sales)
```

```
    res.sq <- residuals**2
    cv.score[i] <- sqrt(sum(res.sq)/model$df.residual)
}
cat(mean(cv.score))
}
cv.fit.lm.3()
```

```
## 0.5734476
```

```
cv.fit.lm.4 <- function (){
#Randomly shuffle the data
ads.copy <- ads[sample(nrow(ads)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(ads.copy)),breaks=10,labels=FALSE)
#Perform 10 fold cross validation
cv.score <- rep(NA, 10)
for(i in 1:10){
    #Segement your data by fold using the which() function
    testIndexes <- which(folds==i, arr.ind=TRUE)
    testData <- ads.copy[testIndexes,]
    trainData <- ads.copy[-testIndexes,]
    model <- glm(sales~TV+radio+newspaper+TV*radio, data=trainData)
    pred <- predict(model, testData, type='response')
    residuals <- (pred - testData$sales)
    res.sq <- residuals**2
    cv.score[i] <- sqrt(sum(res.sq)/model$df.residual)
}
cat(mean(cv.score))
}
cv.fit.lm.4()
```

```
## 0.3073631
```