MATH 4322 Homework 4

Phu Nguyen

Fall 2021

Problem 1

We will review k-fold cross-validation.

(a) Explain how k-fold cross-validation is implemented.

- Data is randomly divided into K subsets or folds of appoximately equal size, n_K.
- The first fold is treated as a validation set.
- The method is fit on the remaining k - 1 folds.
- The mean squared error, MSE, is then computed on the observations in the held-out fold.
- This procedure is repeated k times; each time a different group of observations is treated as a validation set. Which results in k estimates of the test error, $MSE_1, MSE_2, ..., MSE_k$.

(b) What are the advantages and disadvantages of k-fold cross-validation relative to:

i. The validation set approach?

- The validation estimate of the test error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

ii. LOOCV?

- Computational: Doing $LOOCV$ (= K-fold CV for K = n) is tough for computationally intensive models, as apposed to 5 or 10-fold CV, especially for large n.
- The model is fit only $K \ll n$ times.
- K-fold CV doesn't lose in estimation quality to $LOOCV$.
- The variability in K-fold error estimates is negligible.

Problem 2

Suppose that we use some statistical learning method to make a prediction for the response Y for a particular value of the predictor X. Carefully describe how we might estimate the standard deviation of our prediction.

- We can use Bootstrap method. With Bootstrap method we can use the formula $SE(\bar{x}) = \frac{s}{\sqrt(n)} = \sqrt(\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n(n-1)})$

Problem 3

We will perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follows:

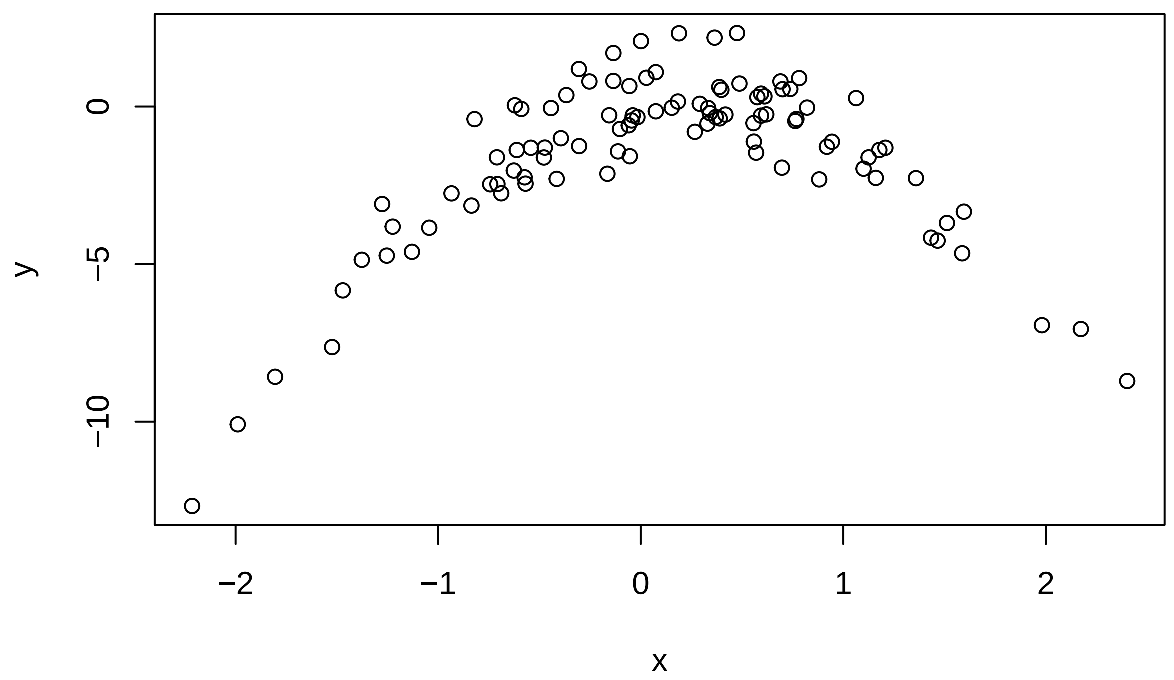
```
set.seed(1)
x=rnorm(100)
y=x-2*x^2+ rnorm(100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

```
library(boot)
dataframe = data.frame(x,y)
model.glm = glm(y ~ x)
cv.model = cv.glm(dataframe, model.glm)
```

(b) Create a scatterplot of X against Y . Comment on what you find.

```
plot(x, y)
```



* It looks like it's quadratic.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

i. $Y = \beta_0 + \beta_1 X + \epsilon$
ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

Note: you might find it helpful to use the `data.frame()` function to create a single data set containing both X and Y.

```
set.seed(10)
Data = data.frame(x,y)
for (i in 1:4) {
  fit.glm = glm(y ~poly(x, i))
  validation = cv.glm(Data, fit.glm)
  print(paste(i, validation$delta[1]))
}
```

```
## [1] "1 7.28816160667281"
## [1] "2 0.937423637615552"
## [1] "3 0.95662183010894"
## [1] "4 0.953904892744804"
```

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
set.seed(100)
Data = data.frame(x,y)
for (i in 1:4) {
  fit.glm = glm(y ~poly(x, i))
  validation = cv.glm(Data, fit.glm)
  print(paste(i, validation$delta[1]))
}
```

```
## [1] "1 7.28816160667281"
## [1] "2 0.937423637615553"
## [1] "3 0.95662183010894"
## [1] "4 0.953904892744804"
```

- We get the same answer because the LOOCV loops through and creates n observations or training sets and gets the average of those errors which leads to more consistent results.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

- The smallest LOOCV error is the second degree poly, this is what I expected because it usually have the highest difference in the MSE.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

Problem 4

We will use a logistic regression to predict the probability of `default` using income and `balance` on the `Default` data set in the ISLR package. 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.

```
library(ISLR)
attach(Default)
set.seed(10)
fit.default.glm = glm(default ~ income + balance, data = Default, family = "binomial")
summary(fit.default.glm)
```

```
##
## 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.

```
train = sample(dim(Default)[1], dim(Default)[1] / 2)
```

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

```
fit.default.glm = glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
summary(fit.default.glm)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4914  -0.1412  -0.0550  -0.0193   3.7586
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.180e+01  6.250e-01 -18.879  < 2e-16 ***
## income       2.323e-05  7.140e-06   3.253  0.00114 **
## balance      5.725e-03  3.237e-04  17.687  < 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: 1463.69  on 4999  degrees of freedom
## Residual deviance:  773.56  on 4997  degrees of freedom
## AIC: 779.56
##
## 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 0.5.

```r
pred = predict(fit.default.glm, newdata = Default[-train, ], type = "response")
pred.glm = rep("No", length(pred))
pred.glm[pred > 0.5] = "Yes"
```

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

```r
mean(pred.glm != Default[-train, ]$default)
```

```
## [1] 0.0288
```

(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.

```r
for (i in 1:3) {
  train = sample(dim(Default)[1], dim(Default)[1] / 2)
  fit.default.glm = glm(default ~ income + balance, data = Default, family = "binomial", subset = train
  pred = predict(fit.default.glm, newdata = Default[-train, ], type = "response")
  pred.glm = rep("No", length(pred))
  pred.glm[pred > 0.5] = "Yes"
  print(mean(pred.glm != Default[-train, ]$default))
}
```

```
## [1] 0.0272
## [1] 0.0288
## [1] 0.026
```

- We can see that we do not get the same value if we don't set the seed. The validation estimate of the test error rate can be variable, depending on precisely which observation are included in the training set and which observation are included in the validation set.

Problem 5

We continue to consider the use of a logistic regression model to predict the probability of `default` using income and `balance` on the `Default` data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the *glm()* function. Do not forget to set a random seed before beginning your analysis.

(a) Using the *summary()* and *glm()* functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

```
set.seed(10)
fit.glm.default = glm(default ~ income + balance, data = Default, family = "binomial")
summary(fit.glm.default)
```

```
##
## 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) Write a function, *boot.fn()*, that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
library(boot)
boot.fn = function(dat, idx) mean(dat[idx],na.rm = TRUE)
```

(c) Use the *boot()* function together with your *boot.fn()* function to estimate the standard errors of the logistic regression coefficients for income and balance.

```
boot.out = boot(data = Default, statistic = boot.fn, R = 1000)
boot.out
```

(d) Comment on the estimated standard errors obtained using the *glm()* function and using your bootstrap function.

- The bootstrap estimates of the standard errors for the coefficients β_0, β_1 and β_2 are respectively 0.4239, 4.583 x 10^(-6) and 2.268 x 10^(-4). The estimated standard errors obtained by the two methods are pretty close.

Problem 6

We will now 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 $\hat{\mu}$.

```
library(MASS)
attach(Boston)
(mu.hat = mean(medv))
```

```
## [1] 22.53281
```

(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.

```
(se.hat = sd(medv) / sqrt(dim(Boston)[1]))
```

```
## [1] 0.4088611
```

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

```
set.seed(10)
boot.fn = function(data, index) {
  mu = mean(data[index])
  return (mu)
}
boot(medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 22.53281 -0.008041502   0.4017124
```

(d) Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of *medv*. Compare it to the results obtained using `t.test(Boston$medv)`. *Hint*: You can approximate a 95% confidence interval using the formula $[\hat{\mu} - 2 \times \mathrm{SE}(\hat{\mu}), \hat{\mu} + 2 \times \mathrm{SE}(\hat{\mu})]$.

```
t.test(medv)
```

```
##
##  One Sample t-test
##
## data:  medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

```
CI.mu = c(22.53 - 2 * 0.4119, 22.53 + 2 * 0.4119)
CI.mu
```

```
## [1] 21.7062 23.3538
```

(e) Based on this data set, provide an estimate, $\hat{\mu}_{\mathrm{med}}$, for the median value of *medv* in the population.

```
med = median(medv)
med
```

```
## [1] 21.2
```

(f) We now would like to estimate the standard error of $\hat{\mu}_{\mathrm{med}}$. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
boot.fn = function(data, index) {
  mu = median(data[index])
  return (mu)
}
boot(medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*     21.2 -0.00885    0.376465
```

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$. (You can use the quantile() function.)

```
percent = quantile(medv, c(0.1))
percent
```

```
##   10%
## 12.75
```

(h) Use the bootstrap to estimate the standard error of $\hat{\mu}_{0.1}$. Comment on your findings.

```
boot.fn = function(data, index) {
  mu = quantile(data[index], c(0.1))
  return (mu)
}
boot(medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75  0.0287   0.4784666
```

- For the 10th percentile we get a value of 12.75 with a relatively love standard error of 0.5113