HW5 Solution CS6220-Data Mining

1. JWHT problem 8, p. 200

(a)

```
set.seed (20)
x=rnorm (100)
y=x-2* x^2+ rnorm (100)
```

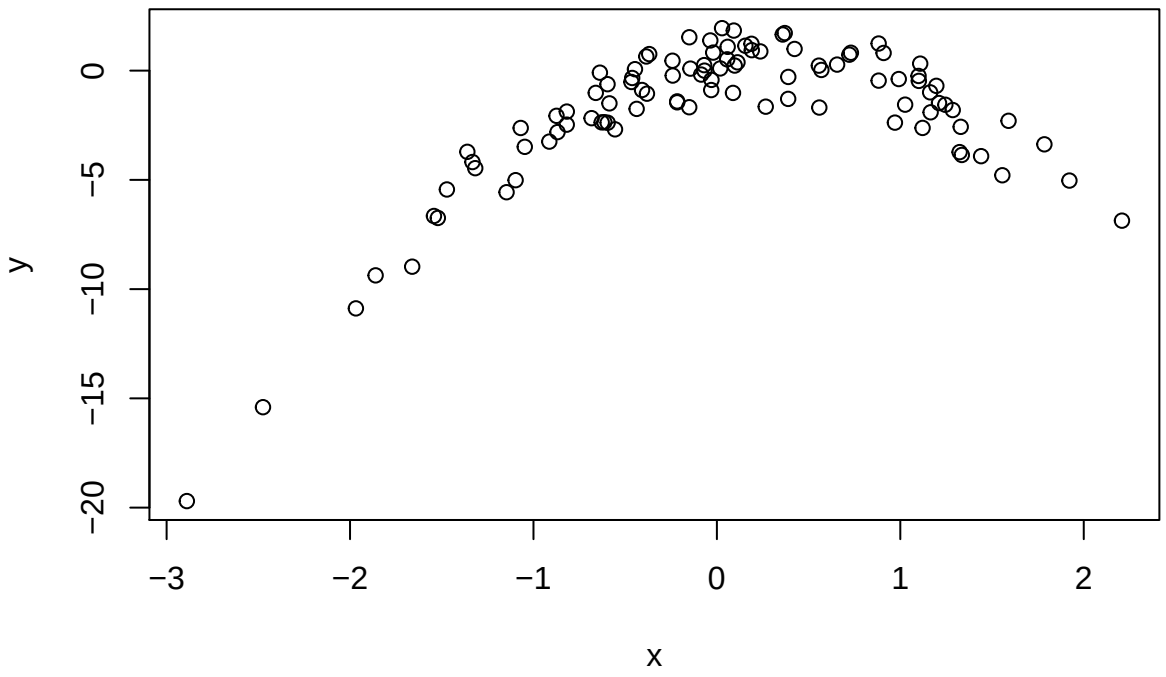
n: 100

p: 2

Equation: $Y_i = X_i - 2X_i^2 + \varepsilon_i$, where ε_i are independent, and follow a Normal distribution with mean 0 and variance 1.

(b)

```
plot(y~x)
```



The relationship between X and Y is clearly curvilinear.

(c)

```
#Create the dataframe from the data
mydata = data.frame(x=x, y=y)

library(boot)

#create an array to save the error for each of suggested models; i to iv
myerror=rep(0,4)
```

```
#We use poly(x,i) for i:1 to 4 for models i to iv
for (i in 1:4)
{
glm.fit = glm(y ~ poly(x,i), data=mydata)
myerror[i] = cv.glm(mydata ,glm.fit)$delta [1]
}
myerror
```

```
## [1] 9.5329671 1.0088648 1.0322236 0.9925797
```

The error of the quadratic model, model ii, is the lowest one. So, it fits the data better than the other ones as expected. The linear model has a worse bias (i.e., it does not accurately represent the true curvilinear fit). The last two models have a worse variance (i.e., they fir flexible non-linear relationships, but overfit the data).

(d)

```
set.seed (25)
x=rnorm (100)
y=x-2* x^2+ rnorm (100)
mydata = data.frame(x=x, y=y)

#calculate the error for each of the four suggested models
for (i in 1:4)
{
glm.fit = glm(y ~ poly(x,i), data=mydata)
myerror[i] = cv.glm(mydata ,glm.fit)$delta [1]
}
myerror
```

```
## [1] 6.5455691 0.8687364 0.8687884 0.8809044
```

Again, the quadratic model has the lowest error. Note that for each model the absolute values of the errors differ from the values in (c). This is due to the random variation in the data. In other words, cross-validated error is itself a random variable

(e)

The best model is the model ii that error is reduced considerably compared to the error of model i. The model iii and iv not improved clearly compared to model ii.

(f)

```
myglm = glm(y ~ poly(x,4),data = mydata)
summary(myglm)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4), data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8985  -0.6315  -0.1868   0.6759   2.6431
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.25373    0.09241 -24.387   <2e-16 ***
## poly(x, 4)1  10.52646    0.92413  11.391   <2e-16 ***
## poly(x, 4)2 -22.86478    0.92413 -24.742   <2e-16 ***
## poly(x, 4)3  -1.06038    0.92413  -1.147    0.254
## poly(x, 4)4  -0.28891    0.92413  -0.313    0.755
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.8540232)
##
##     Null deviance: 715.944  on 99  degrees of freedom
## Residual deviance:  81.132  on 95  degrees of freedom
## AIC: 274.88
##
## Number of Fisher Scoring iterations: 2
```

The p-values show the significance of x and x^2 and not other terms in the model.

2. JWHT problem 9, p. 201, but for the surgical unit dataset given as an example in the class. Use the variable lsurv.

```
setwd('/Users/ovitek/Dropbox/Olga/Teaching/CS6220/Fall15/Homeworks/Hw5')
sdata <- read.table("surgical.txt")
dimnames(sdata)[[2]] <- c('blood', 'prog', 'enz', 'liver', 'age', 'female', 'modAlc', 'heavyAlc', 'surv
# Check for NA values
sum(is.na(sdata))
```

```
## [1] 0
```

```
attach(sdata)
```

(a)

```
#an estimate for the population mean
mu = mean(lsurv)
mu
```

```
## [1] 6.430481
```

(b)

```
#an estimate of the standard error
sd = sd(lsurv)/sqrt(length(lsurv))
sd
```

```
## [1] 0.06689616
```

(c)

```r
#estimate the standard error of mu using the bootstrap
#Create the statistic function for boot() method
myfunction = function(data, index) return(mean(data[index]))
#Call the library(boot) if you have not yet
bstrap = boot(lsurv, myfunction, 1000)
bstrap
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = lsurv, statistic = myfunction, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 6.430481 0.001171352  0.06677149
```

As we see the result of bootstrap is pretty close to the actual result.

(d)

```r
#t.test
t.test(lsurv)
```

```
##
##  One Sample t-test
##
## data:  lsurv
## t = 96.126, df = 53, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  6.296305 6.564658
## sample estimates:
## mean of x
##  6.430481
```

```r
#95% confidence interval for the mean of lsurv
c(mu - 2 *0.06677149, mu + 2 *0.06677149)
```

```
## [1] 6.296939 6.564024
```

The result of confidence interval and the t.test are very close.

(e)

```r
#an estimate of mumed
mumed = median(lsurv)
mumed
```

```
## [1] 6.406
```

(f)

```
#estimate the standard error of mumed
#statistic function for boot()
myfunction2 = function(data, index) return(median(data[index]))

bstrap = boot(lsurv, myfunction2, 1000)
bstrap
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = lsurv, statistic = myfunction2, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    6.406 0.010907  0.05631268
```

The standard error is very small compared to median value.

(g)

```
#an estimate for the tenth percentile of lsurv
mu.01 = quantile(lsurv, c(0.1))
mu.01
```

```
##    10%
## 5.8711
```

(h)

```
#estimate the standard error of mu.01
#statistic function for boot()
myfunction3 = function(data, index) return(quantile((data[index]),c(0.1)))

bstrap = boot(lsurv, myfunction3, 1000)
bstrap
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = lsurv, statistic = myfunction3, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   5.8711 0.0137097   0.1122078
```

The standard error is again very small compared to 10% value.

Also, note that the standard error for the 10th quantile is larger than the standard error for the median. This is a typical pattern in bootstrap, reflecting the fact that the tail quantiles are more difficult to estimate than the center of the distribtion.

3. JWHT problem 9, p. 263, but for the surgical unit dataset given as an example in the class.Skip (e) & (f)

(a)

```
set.seed(20)
setwd('/Users/ovitek/Dropbox/Olga/Teaching/CS6220/Fall15/Homeworks/Hw5')
sdata <- read.table("surgical.txt")
dimnames(sdata)[[2]] <- c('blood', 'prog', 'enz', 'liver', 'age', 'female', 'modAlc', 'heavyAlc', 'surv
# Check for NA values
sum(is.na(sdata))
```

```
## [1] 0
```

```
attach(sdata)
```

```
## The following objects are masked from sdata (pos = 3):
##
##     age, blood, enz, female, heavyAlc, liver, lsurv, modAlc, prog,
##     surv
```

```
#remove the surve column
mydata = sdata[, -9]
#index of test samples
test = sample(1:nrow(mydata), nrow(mydata)/10)
#Test set
testset = mydata[test,]
#Train set
trainset = mydata[-test,]
```

(b)

```
library(ISLR)
#Fit and evaluate linear model
mylm = lm(lsurv~., trainset)
mypredict = predict(mylm, testset)
#Test error
mean((testset[, "lsurv"] - mypredict)^2)
```

```
## [1] 0.05418014
```

(c)

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-2
```

```r
#Create the matrices for cv.glmnet()
train.mat = model.matrix(lsurv~., trainset)
test.mat = model.matrix(lsurv~., testset)
```

Ridge:

```r
grid = 10 ^ seq(10, -2, length=100)
#alpha=0 the ridge penalty
ridge.mod = cv.glmnet(train.mat, trainset[,"lsurv"], alpha=0, lambda=grid, thresh=1e-12)
#Finding lambda
ridge.lambda = ridge.mod$lambda.min
ridge.lambda
```

```
## [1] 0.01
```

```r
#prediction
ridge.pred = predict(ridge.mod, newx=test.mat, s=ridge.lambda)
#Test error
mean((testset[, "lsurv"] - ridge.pred)^2)
```

```
## [1] 0.05034646
```

The error of ridge method is slightly less than lm method.

(d)

Lasso:

```r
#alpha=1 is the lasso penalty
lasso.mod = cv.glmnet(train.mat, trainset[,"lsurv"], alpha=1, lambda=grid, thresh=1e-12)
#Finding lambda
lasso.lambda = lasso.mod$lambda.min
lasso.lambda
```

```
## [1] 0.01
```

```r
#prediction
lasso.pred = predict(lasso.mod, newx=test.mat, s=lasso.lambda)
#Test error
mean((testset[, "lsurv"] - lasso.pred)^2)
```

```
## [1] 0.03944702
```

(g)

The error of Lasso is even less than the Ridge model

Coefficients:

```r
coef(lasso.mod)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)  4.2646018753
## (Intercept)  .
## blood        0.0358397204
## prog         0.0121890865
## enz          0.0127416679
## liver        0.0591138590
## age         -0.0009053423
## female       0.0508318609
## modAlc       .
## heavyAlc     0.3666132442
```

There are seven non-zero coefficients besides intercept.