STAT 8330 FALL 2015 ASSIGNMENT 2

Peng Shao

September 15, 2015

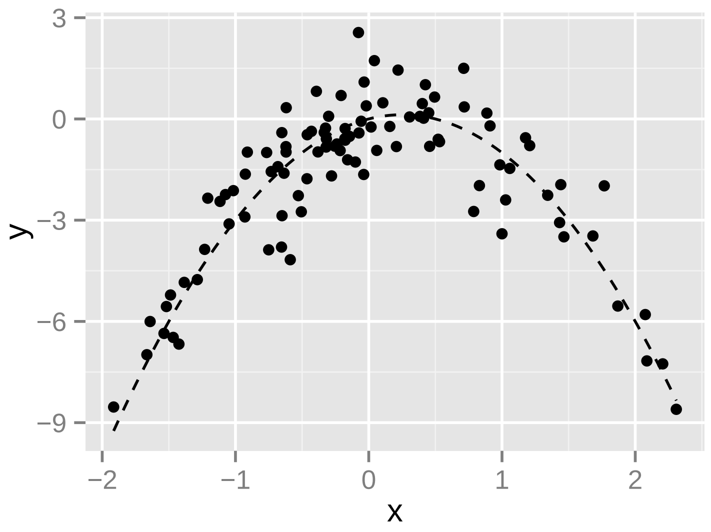
▶ **Exercises 5.8.** **Solution.**

(a). $n = 100$ and $p = 2$. The model is

$$Y_i = X_i - 2X_i^2 + \varepsilon_i$$

(b). Obivously, the scatterplot has a quadratic pattern of x.

```
#b
data.xy <- data.frame(x, y)
ggplot(data = data.xy, aes(x = x, y = y)) +
    geom_point() +
    stat_function(fun = fun.y, linetype = 2)
```



(c).

```
#c
set.seed(2)
cv.error <- rep(0, 4)
for (i in 1:4){
    glm.fit <- glm(y ~ poly(x,i), data = data.xy)
    cv.error[i] <- cv.glm(data.xy, glm.fit)$delta[1]
}
cv.error
```

```
## [1] 5.890979 1.086596 1.102585 1.114772
```

(d). The results are same because LOOCV does not use any random samping. It evalutes every observation.

```
#d
set.seed(3)
cv.error2 <- rep(0, 4)
for (i in 1:4){
    glm.fit <- glm(y ~ poly(x,i), data = data.xy)
    cv.error2[i] <- cv.glm(data.xy, glm.fit)$delta[1]
}
cv.error2
```

```
## [1] 5.890979 1.086596 1.102585 1.114772
```

(e). The model which has lowerest LOOCV is the cubic model, which is not what we expect. On one hand, this may be cause by the random error, which makes the quadratic pattern not that obviously. On the other hand, the LOOCV is very close to the MSE, and the model fitting method is to minimize the MSE of training set, so there may be some overfitting problems. Actually, if we try 5-fold or 10-fold cross-validation, we can see that the quadratic model always has the lowest CV. Furthermore, as the value of K increases, the CV of cubic is decreasing, so we can believe that when $K = n$, the the CV of cubic is the smallest for all K.

(f). It is easily to see that, for all these four models, only intercept, first order term and quadratic term are significatn. This is different from (c)., but this is what we expect since we create the data like this way.

```
## [[1]]
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -1.827707  0.2362206 -7.7372898 9.181461e-12
## poly(x, i)   2.316401  2.3622062  0.9806091 3.292002e-01
##
## [[2]]
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  -1.827707  0.1032351 -17.70431 3.804657e-32
## poly(x, i)1   2.316401  1.0323515   2.24381 2.711854e-02
## poly(x, i)2 -21.058587  1.0323515 -20.39866 7.333860e-37
##
## [[3]]
##                 Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  -1.8277074  0.1037248 -17.6207390 7.610579e-32
## poly(x, i)1   2.3164010  1.0372479   2.2332183 2.785714e-02
## poly(x, i)2 -21.0585869  1.0372479 -20.3023667 1.636959e-36
## poly(x, i)3  -0.3048398  1.0372479  -0.2938929 7.694742e-01
##
## [[4]]
##                 Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  -1.8277074  0.1041467 -17.5493533 1.444977e-31
## poly(x, i)1   2.3164010  1.0414671   2.2241711 2.850549e-02
## poly(x, i)2 -21.0585869  1.0414671 -20.2201171 3.457023e-36
## poly(x, i)3  -0.3048398  1.0414671  -0.2927023 7.703881e-01
## poly(x, i)4  -0.4926249  1.0414671  -0.4730105 6.372907e-01
```

▶ **Exercises 5.8. Solution.**

(a). The mean is $\hat{\mu} = 22.5328063$.

(b). The standard error of mean is $SE\{\hat{\mu}\} = 0.4088611$.

(c). The standard error of mean using bootstrap is $SE_{bootstrap}\{\hat{\mu}\} = 0.4119$, which is very close to the estimate found in (b). of 0.4089.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = mean.fn, R = 1000)
```

```
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 22.5328063  8.517589e-03  0.41193744
## t2*  0.4088611 -8.774353e-05  0.01657105
```

(d). The two confidence intervals are listed as below. They are very close.

```
ci.t
```

```
## [1] 21.72953 23.33608
```

```
ci.boot
```

```
## [1] 21.71691 23.33167
```

(e). The estimate of median is $\hat{\mu}_{med} = 21.2$.

(f). The estimate standard error of median is 0.3874004, which is smaller than that of mean. Maybe because the estimator of median is a more robust estimator, and the median is a better parameter to describe the center of the population because of the stablity.

```
#f
median.fn <- function(data, index)
    return(median(data[index]))
median.boot <- boot(medv, median.fn, R = 1000)
median.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = median.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0098   0.3874004
```

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

(g). The estimate of tenth percentile is $\hat{\mu}_{0.1} = 12.75$.

(h). The estimate standard error of tenth percentile is 0.5113487, which is greater than that of mean. It seems reasonable since tenth percentile contains more information from smaller observations and less infomation from larger observations, while the bootstrap will treat all observations as equally likely to estimate the parameter, which may introduce more variation for estimating the skewed parameter. Remind that the median and the mean summary the information of all observations with equal probability, especially then median. So there is no wonder that the estimate of median has the smallest standard error.

```
#h
q.10.fn <- function(data, index)
    return(quantile(data[index], probs = 0.1))
q.10.boot <- boot(medv, q.10.fn, R = 1000)
q.10.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
##
##
## Call:
## boot(data = medv, statistic = q.10.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.00515   0.5113487
```

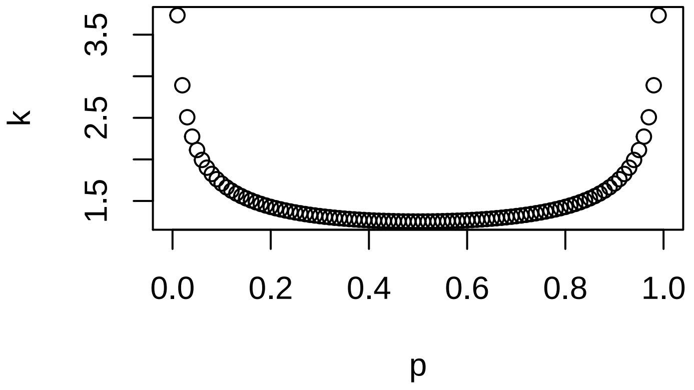
Actually, we know that the stardard error of the estimate of mean is

$$SE\{\hat{\mu}\} = \frac{s}{\sqrt{n}}$$

where s is the standard deviation of the sample. And for large sample, the stardard error of the estimate of pth percentile is

$$SE\{\hat{\mu}_p\} = \frac{s\sqrt{p(1-p)}}{\sqrt{n}\phi(\Phi^{-1}(p))} = SE\{\hat{\mu}\}\frac{\sqrt{p(1-p)}}{\phi(\Phi^{-1}(p))}$$

where ϕ is the pdf of normal distribution and Φ^{-1} is the quantile function of normal distribution. So there is a ratio between the stardard error of the estimate of mean and that of pth percentile. The ratio has a smallest value at $p = 0.5$, which means the median. When going to each boundary, this ratio becomes very large, which implies the high variance of tenth percentile. The plot of the ratio is like this:

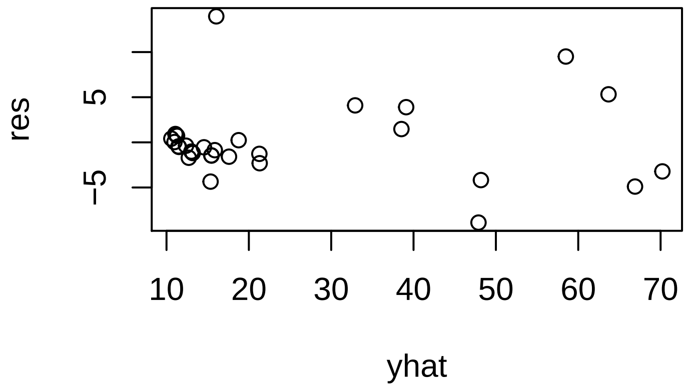


▶ **Problem 3. Solution.**

For this linear regression problem, if we only consider the three predictors and the interactions among them (because when we want to try any other forms of predictor, it means we also need to estimate the form of model, which would be better by nonparametric method), the total number of all possible models will be $2^{2^3-1} - 1 = 127$. So it is possible for us to use exhaustive enumeration method to compare all models to select the "best" model based on CV.

```
##
## Attaching package: 'car'
##
## The following object is masked from 'package:boot':
##
##     logit
```

```
regressor <- c(names(pages)[3],
               names(pages)[4],
               names(pages)[5],
               paste(names(pages)[3],
                     names(pages)[4],
                     sep = "*"),
               paste(names(pages)[3],
                     names(pages)[5],
                     sep = "*"),
               paste(names(pages)[4],
                     names(pages)[5],
                     sep = "*"),
               paste(names(pages)[3],
                     names(pages)[4],
                     names(pages)[5],
                     sep = "*"))
indices <- expand.grid(c(TRUE, FALSE),
                       c(TRUE, FALSE),
                       c(TRUE, FALSE),
                       c(TRUE, FALSE),
                       c(TRUE, FALSE),
                       c(TRUE, FALSE),
                       c(TRUE, FALSE)
                       )
cvError <- rep(NA, 127)
for (n in 1:127){
    lmformula <- reformulate(regressor[as.logical(indices[n, ])],
                             response = "pages")
    glm.fit <- glm(lmformula, data = pages)
    cvError[n] <- cv.glm(data = pages,
                         glmfit = glm.fit,
                         K = 5)$delta[2]
    names(cvError)[n] <- paste(regressor[as.logical(indices[n, ])],
                               collapse = "+")
}
best <- cvError[which.min(cvError)]
best.lm <- lm(as.formula(paste("pages", names(best), sep = "~")),
              data = pages)
res <- resid(best.lm)
yhat <- predict(best.lm)
plot(yhat, res)
```



```
outlierTest(best.lm)
```

```
##   rstudent unadjusted p-value Bonferonni p
## 7 3.870018         0.00077646      0.02407
```

After running the code, we can get the best mode is "pages cl+cl*il+cl*ld+il*ld" with the CV=31.4730839. Since we have 127 models, which too many to display, so I only show the best 5 model and the worst 5 model.

```
options(width = 60)
cvError.sorted <- sort(cvError)
head(cvError.sorted, n = 5)
```

```
##    cl+cl*il+cl*ld+il*ld                    il+ld
##                31.47308                 40.31026
## il+ld+cl*il+cl*ld+il*ld              il+ld+il*ld
##                41.24556                 43.81768
##                   il*ld
##                44.96505
```

```
tail(cvError.sorted, n = 5)
```

```
##                ld+cl*il*ld       il+ld+cl*ld+cl*il*ld
##                   246.3207                   312.7655
##          il+cl*ld+cl*il*ld    ld+cl*il+cl*ld+cl*il*ld
##                   395.6746                  1990.7422
## cl+il+cl*il+cl*ld+cl*il*ld
##                  2513.6995
```

► **Problem 4. Solution.**

It is very similar to problem 3., the main difference is the cost funtion. The cost function for a classification problem is usually the error rate of prediction. So the best model is the model which has smallest error rate.

```
#4
set.seed(1)
des <- read.table(file = "./des_site1and2sp_2.dat",
                  col.names = c("glu",
                                "nud",
                                "utmn",
                                "utme",
                                "sw",
                                "elevation",
                                "slope",
                                "geology",
                                "LTA",
                                "ELT",
                                "site",
                                "subplot"))
attach(des)
glu <- as.factor(glu)
geology <- as.factor(geology)
LTA <- as.factor(LTA)
ELT <- as.factor(ELT)
site <- as.factor(site)
costFunction <- function(y, yhat) return(mean(y != (yhat > 0.5)))
cvError.inter <- rep(NA, 15)
cvError.noint <- rep(NA, 15)
n <- 1
for (i in 5:9){
    for (j in (i+1):10){
        glmformula.inter <- as.formula(
            paste("glu~(",
                  paste(names(des)[c(i, j)],
                        collapse = "+"),
                  ")^2"
            )
        )
        glmformula.noint <- as.formula(
            paste("glu~",
                  paste(names(des)[c(i, j)],
                        collapse = "+")
            )
        )
        logis.fit.inter <- glm(glmformula.inter,
                               family = "binomial",
                               data = des)
        logis.fit.noint <- glm(glmformula.noint,
                               family = "binomial",
                               data = des)
        cvError.inter[n] <- cv.glm(data = des,
                                   glmfit = logis.fit.inter,
                                   cost = costFunction,
                                   K = 10)$delta[1]
        cvError.noint[n] <- cv.glm(data = des,
                                   glmfit = logis.fit.noint,
                                   cost = costFunction,
                                   K = 10)$delta[1]
        names(cvError.inter)[n] <- paste(
            as.character(glmformula.inter)[2],
            as.character(glmformula.inter)[1],
            as.character(glmformula.inter)[3],
            collapse = " ")
        names(cvError.noint)[n] <- paste(
            as.character(glmformula.noint)[2],
            as.character(glmformula.noint)[1],
            as.character(glmformula.noint)[3],
            collapse = " ")
        n <- n + 1
    }
}
best.inter <- min(cvError.inter)
best.noint <- min(cvError.noint)
names(best.inter) <- names(cvError.inter)[which.min(cvError.inter)]
names(best.noint) <- names(cvError.noint)[which.min(cvError.inter)]
```

For 15 models with no interaction, the CVs are

```
best.noint
```

```
## glu ~ slope + geology
##             0.2416107
```

So the best model is "glu~ glu ~ slope + geology", with CV=0.2416107.