Zhangsheng Lai
1002554

# Statistics: Homework 4

1. (a) Given $p_i$ and $q_i$ denote the probability of choosing box 1 and 2 respectively if the ball color chosen is $i$ where $i = \{B, W, G\}$, denoting the three different colors. With the given information of the number of different color balls in the different boxes,

$$\begin{array}{lll} \mathbb{P}(B|1) = 4/10 & \mathbb{P}(B|2) = 2/10 & \mathbb{P}(B|3) = 4/10 \\ \mathbb{P}(W|1) = 3/10 & \mathbb{P}(W|2) = 6/10 & \mathbb{P}(W|3) = 1/10 \\ \mathbb{P}(G|1) = 2/10 & \mathbb{P}(G|2) = 0 & \mathbb{P}(G|3) = 8/10 \end{array}$$

The risk function is represented by,

$$\begin{aligned} R(\theta, \hat{\theta}_{p,q}) &= \mathbb{E}_\theta(|\theta^2 - \hat{\theta}_{p,q}|^2) \\ &= \mathbb{E}_\theta \left( \sum_{i \in \{B,W,G\}} L(\theta, \hat{\theta}_{p,q}(i)) \mathbb{P}(i|\theta) \right) \end{aligned}$$

where $L(\theta, \hat{\theta}_{p,q}(i)) = L(\theta, 1)p_i + L(\theta, 2)q_i + L(\theta, 3)(1 - p_i - q_i)$.Therefore,

$$\begin{aligned} R(1, \hat{\theta}_{p,q}) &= [q_B + 4(1 - p_B - q_B)]\frac{4}{10} + [q_W + 4(1 - p_W - q_W)]\frac{3}{10} + [q_G + 4(1 - p_G - q_G)]\frac{2}{10} \\ R(2, \hat{\theta}_{p,q}) &= [9p_B + 4q_B + 49(1 - p_B - q_B)]\frac{2}{10} + [9p_W + 4q_W + 49(1 - p_W - q_W)]\frac{6}{10} \end{aligned}$$

(b) Bayes risk is given by

$$r(f, \theta) = \int R(\theta, \hat{\theta}_{p,q}) f(\theta)\, d\theta$$

but since our scenario is discrete, we instead have

$$\begin{aligned} r(f, \theta) &= \sum_{\theta=1,2} R(\theta, \hat{\theta}_{p,q}) \mathbb{P}(\theta) \\ &= \lambda R(1, \hat{\theta}_{p,q}) + (1 - \lambda) R(2, \hat{\theta}_{p,q}) \end{aligned}$$

where $R(1, \hat{\theta}_{p,q})$ and $R(2, \hat{\theta}_{p,q})$ are the values are from (a).

(c) Given $\lambda = 1/2$, we have

$$r(f, \theta) = \frac{1}{2}\left(R(1, \hat{\theta}_{p,q}) + R(2, \hat{\theta}_{p,q})\right) = \frac{1}{20}(428 - 96p_B - 102q_B - 252p_W - 279q_W - 8p_G - 6q_G)$$

thus to the infimum of Bayes risk is when $q_B = q_W = p_G = 1$.

2.

```
library(leaps)
library(dplyr)

# Read csv file into dataframe car
car <- read.csv('carmpgdat.csv')
```

3.

```
library(dplyr)
library(magrittr)


# Reading data with separator tab
raw_riasec <- read.csv('RIASEC.csv',sep = '\t')

# (a)CLEANING UP
# List of realistic traits
realistic_trait <- c('R1','R2','R3','R4','R5','R6','R7','R8')

# Extracting out the realistic traits
raw_realistic <- raw_riasec[,realistic_trait]

# Removing rows with -1 from the dataframe
realistic <- raw_realistic %>%
  filter(R1* R2 * R3 * R4 * R5 * R6 * R7 * R8 > 0)


# (b)MODEL SELECTION

# Computing the score for the R trait
realistic <- realistic %>%
  rowwise() %>%
  mutate(Rscore = mean(c(R1, R2, R3, R4, R5, R6, R7, R8)))

# Generating the training and validation set
tr_realistic <- realistic[1:6500,]
val_realistic <- realistic[-(1:6500),]

# Building the linear model
lm <- lm(Rscore~R1, data = tr_realistic)
avg_RSS_tr = mean(lm$residuals^2)

# estimated regression function and residual sum of squares
print(lm$coefficients)
# R1 = 1.0011862 + 0.4282061 * R1


# (c)VALIDATION
reg.fn <- function(x) 1.0011862 + 0.4282061 * x

val_realistic %<>%
  mutate(pred_Rscore = reg.fn(R1),
         residuals = reg.fn(R1) - Rscore )

avg_RSS_val = mean(val_realistic$residuals^2)

print (avg_RSS_tr) # 0.4540176
print (avg_RSS_val) # 0.5376852

# The residual sum of squares for the validation set using the regression function
# is larger than the residual sum of square for the training set but are of the
# same order. Thus the model generalizes well.
```

4. (a) Using basic Monte Carlo,

$$I = \int_1^2 f(x|1.5, 2.3)\, dx = \frac{1}{N}\sum_{i=1}^{N} f(X_i|1.5, 2.3)$$

```
def mc_integrate(alpha, beta, N = 100000):
    sample = np.random.uniform(1,2, size = (1, N))
    return np.mean([gamma.pdf(x, alpha, loc = 0, scale = beta) for x in sample])
mc_integrate(1.5, 2.3)
```

Estimate $I$ using Monte Carlo method is 0.20449041416849226.

```
# Empirical distribution to draw bootstrap samples
N = 100000
sample = np.random.uniform(1,2, size = (1, N))
emp_dist = [gamma.pdf(x, 1.5, loc = 0, scale = 2.3) for x in sample]
```

```
# Creating the 10000 bootstrap samples
bs_samples = [np.random.choice(emp_dist[0], 100000) for x in range(10000)]
# Getting an estimate of I for each bootstrap sample
bs_estimates = [np.mean(bs_samples[i]) for i in range(10000)]
# Computing the standard error obtained using bootstrap method
mean_bs_estimates = np.mean(bs_estimates)
bs_se = np.mean([(mean_bs_estimates - bs_estimates[i])**2 for i in range(10000)])

# Using the Monte Carlo method
# Resampling the distribution 10000 times
mc_estimates = [mc_integrate(1.5, 2.3) for i in range(10000)]
# Computing the standard error obtained from MC method
mean_mc_estimates = np.mean(mc_estimates)
mc_se = np.mean([(mean_mc_estimates - mc_estimates[i])**2 for i in range(10000)])
```

Bootstrap method gives a standard error of 3.23599044314e-10 and MC method gives a standard error of 3.32551091852e-10.

(b) The standard error of