Exam Solution

Ahmed Alhasan

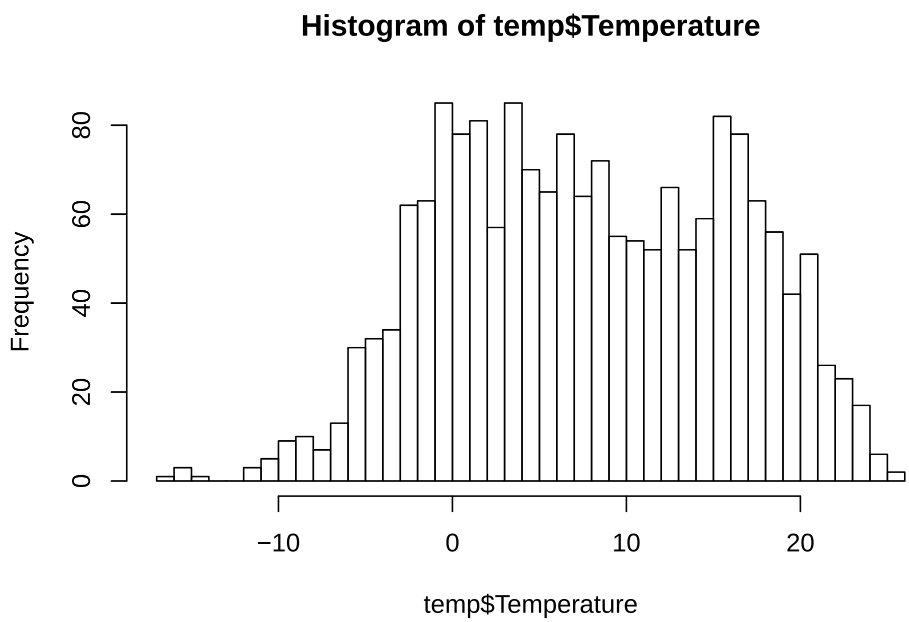
3/19/2020

I solemnly swear that I wrote the exam honestly, I did not use any unpermitted aids, nor did I communicate with anybody except of the course examiners.

Ahmed Alhasan

Assignment 1

```
RNGversion("3.5.2")
## Assignment 1

temp <- read.csv2("Dailytemperature.csv")
phis <- matrix(0,1792,202)
temp <- cbind(temp,phis)


i <- -50
for(j in 3:103){
  for(obs in 1:1792){
    temp[obs,j]     <- sin(0.5^i * temp$Day[obs])
    temp[obs,j+101] <- cos(0.5^i * temp$Day[obs])
  }
  i <- i + 1
}

hist(temp$Temperature, breaks = 50)
```



- From the histogram we can see Temperature is fairly follow a normal distribution

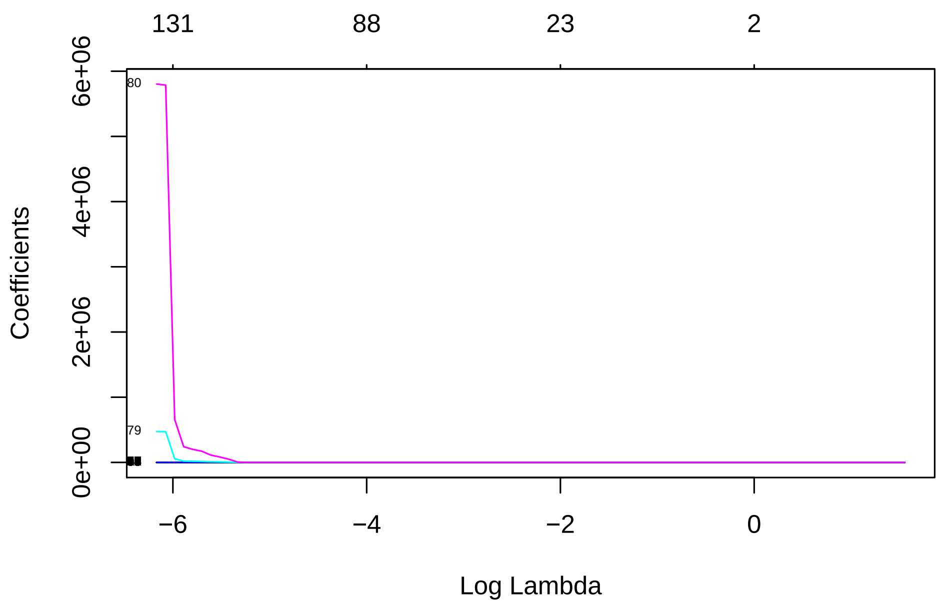
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

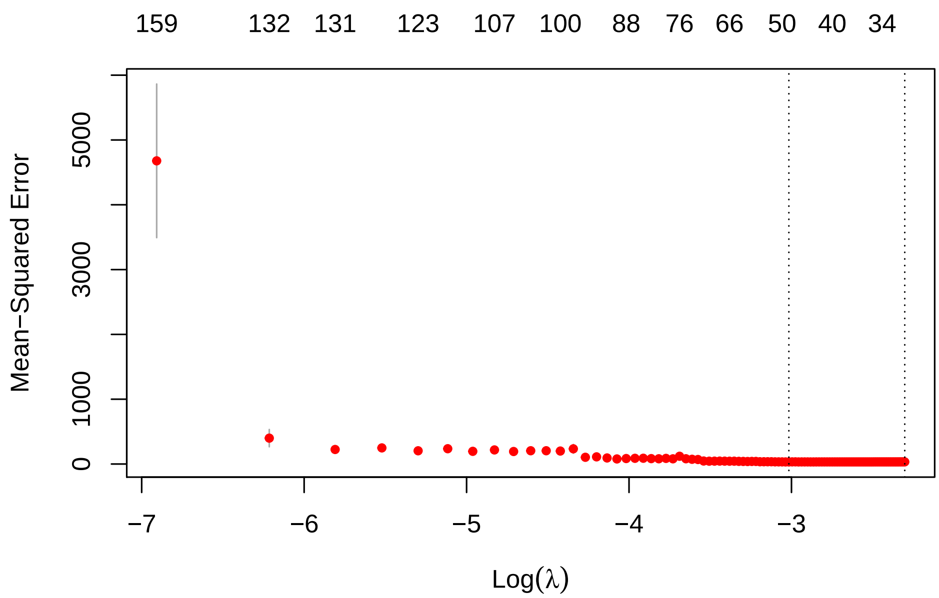
```
lasso <- glmnet(x = as.matrix(temp[,-c(1,2)]),
                y = as.matrix(temp$Temperature),
                alpha = 1,
                family = "gaussian")

plot(lasso, xvar="lambda", label=TRUE)
```

2p



- By increasing the penalty factor we eliminate the least correlated predictors, there is sharp decrease in the number of predictors once we start increasing the penalty factor, until it reaches 2 predictors when loglambda = 0

```
set.seed(12345)
lasso_cv <- cv.glmnet(x = as.matrix(temp[,-c(1,2)]),
                      y = as.matrix(temp$Temperature),
                      alpha=1,
                      family="gaussian",
                      lambda = 0:100 * 0.001)

plot(lasso_cv)
```




```
c("Minimum Lambda" = lasso_cv$lambda.min)
```

```
## Minimum Lambda
##          0.049
```

```
c("1se Lambda" = lasso_cv$lambda.1se)
```

```
## 1se Lambda
##        0.1
```

- The obtimal lambda (lambda.min) is statistical significant at alpha = 1-1sd than log-lambda = -4 since we can see the the two dotted lines on the plot represent lambda.min and lambda at 1 standard deviation.

```
lasso_opt <- glmnet(x = as.matrix(temp[,-c(1,2)]),
                    y = as.matrix(temp$Temperature),
                    alpha = 1,
                    family = "gaussian",
                    lambda = lasso_cv$lambda.min)
c("Number of non-zero Features" = lasso_opt$df)
```

```
## Number of non-zero Features
##                          48
```

```
coef(lasso_opt, s = lasso_cv$lambda.min)
```

```
## 203 x 1 sparse Matrix of class "dgCMatrix"
##                            1
```

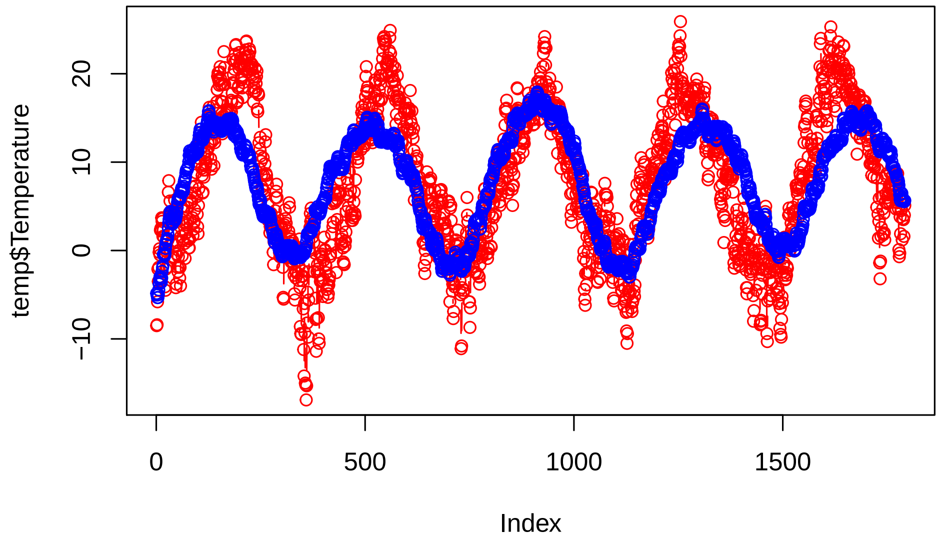
```
## (Intercept) -12.586014923
## 1              .
## 2            -0.005472565
## 3              .
## 4              .
## 5              .
## 6              .
## 7             0.157510067
## 8              .
## 9              .
## 10             .
## 11             .
## 12             .
## 13            0.023906747
## 14             .
## 15             .
## 16             .
## 17             .
## 18             .
## 19             .
## 20             .
## 21            0.039367414
## 22             .
## 23             .
## 24           -0.066441656
## 25             .
## 26             .
## 27             .
## 28             .
## 29             .
## 30             .
## 31            0.121613825
## 32           -0.105555009
## 33             .
## 34           -0.061832520
## 35             .
## 36             .
## 37             .
## 38             .
## 39             .
## 40            0.047146648
## 41           -0.004487736
## 42             .
## 43             .
## 44             .
## 45            0.011181097
## 46             .
## 47             .
## 48             .
## 49             .
## 50             .
## 51            0.122014178
## 52           -0.121118018
## 53             .
```

```
## 54          -0.387043786
## 55           0.254979573
## 56           0.666863121
## 57           7.461962911
## 58           2.430565813
## 59           3.969918103
## 60          19.260811784
## 61          12.005850798
## 62           .
## 63           .
## 64           .
## 65           .
## 66           .
## 67           .
## 68           .
## 69           .
## 70           .
## 71           .
## 72           .
## 73           .
## 74           .
## 75           .
## 76           .
## 77           .
## 78           .
## 79           .
## 80           .
## 81           .
## 82           .
## 83           .
## 84           .
## 85           .
## 86           .
## 87           .
## 88           .
## 89           .
## 90           .
## 91           .
## 92           .
## 93           .
## 94           .
## 95           .
## 96           .
## 97           .
## 98           .
## 99           .
## 100          .
## 101          .
## 102          .
## 103         -0.364314295
## 104          0.095882284
## 105          .
## 106          .
## 107          0.107687691
```

```
## 108             .
## 109             .
## 110             .
## 111             .
## 112             .
## 113            -0.009547945
## 114             .
## 115             .
## 116             0.005829410
## 117             .
## 118             .
## 119             0.015257004
## 120             .
## 121             0.049882913
## 122             0.144302688
## 123             .
## 124             .
## 125             .
## 126             .
## 127             .
## 128             0.116765959
## 129             0.014188369
## 130             0.031373228
## 131             0.275418442
## 132             0.237256685
## 133            -0.011109409
## 134             .
## 135            -0.042394674
## 136             .
## 137             .
## 138             .
## 139             .
## 140            -0.051312885
## 141             .
## 142             0.145125056
## 143             .
## 144             .
## 145             .
## 146             .
## 147            -0.070451157
## 148             .
## 149             0.036864005
## 150             .
## 151             .
## 152            -0.085867271
## 153             .
## 154            -0.065635870
## 155            -0.342290933
## 156             0.142477563
## 157             0.707810361
## 158            -3.948318908
## 159             1.725748910
## 160             8.690409989
## 161             .
```

```
## 162           .
## 163           .
## 164           .
## 165           .
## 166           .
## 167           .
## 168           .
## 169           .
## 170           .
## 171           .
## 172           .
## 173           .
## 174           .
## 175           .
## 176           .
## 177           .
## 178           .
## 179           .
## 180           .
## 181           .
## 182           .
## 183           .
## 184           .
## 185           .
## 186           .
## 187           .
## 188           .
## 189           .
## 190           .
## 191           .
## 192           .
## 193           .
## 194           .
## 195           .
## 196           .
## 197           .
## 198           .
## 199           .
## 200           .
## 201           .
## 202           .
```

```r
pred <- predict(lasso_opt, newx = as.matrix(temp[,-c(1,2)]))

mydata <- data.frame(x = temp$Day, y = temp$Temperature, yhat = pred)

plot(temp$Temperature, type = "b", col = "red")
points(pred, type = "b", col = "blue")
```



- The predicted values give a fairly good approximation to the original temperature

Assignment 2

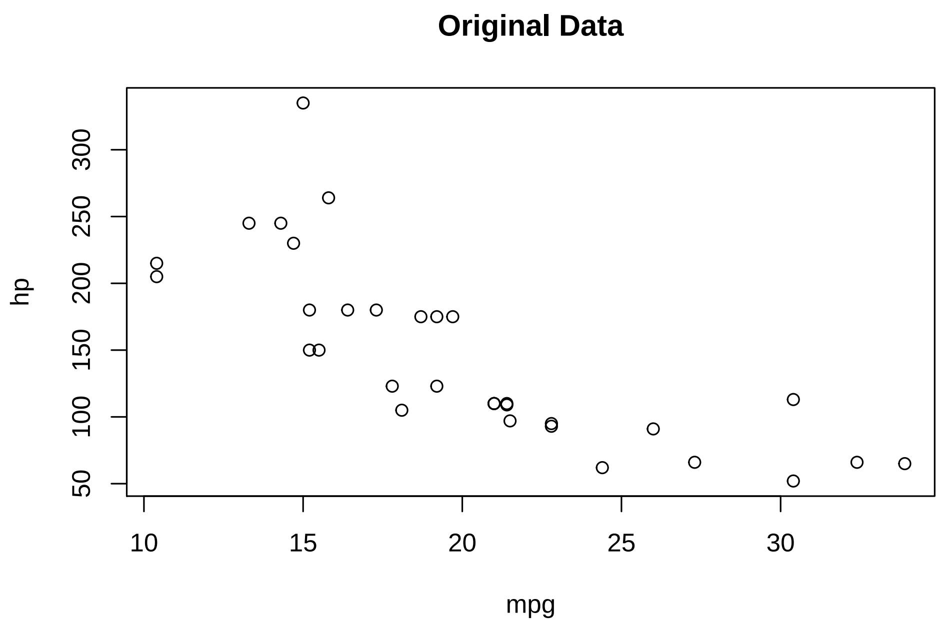
1)

```
data(mtcars)
cars <- mtcars[,c(1,4)]
Var1  <- var(cars)

comps <- eigen(Var1)$vectors
colnames(comps) <- c("PC1", "PC2")

c("First Component" = comps[,1])
```

```
## First Component1 First Component2
##      -0.06827783       0.99766635
```

```
plot(cars, main = "Original Data")
```

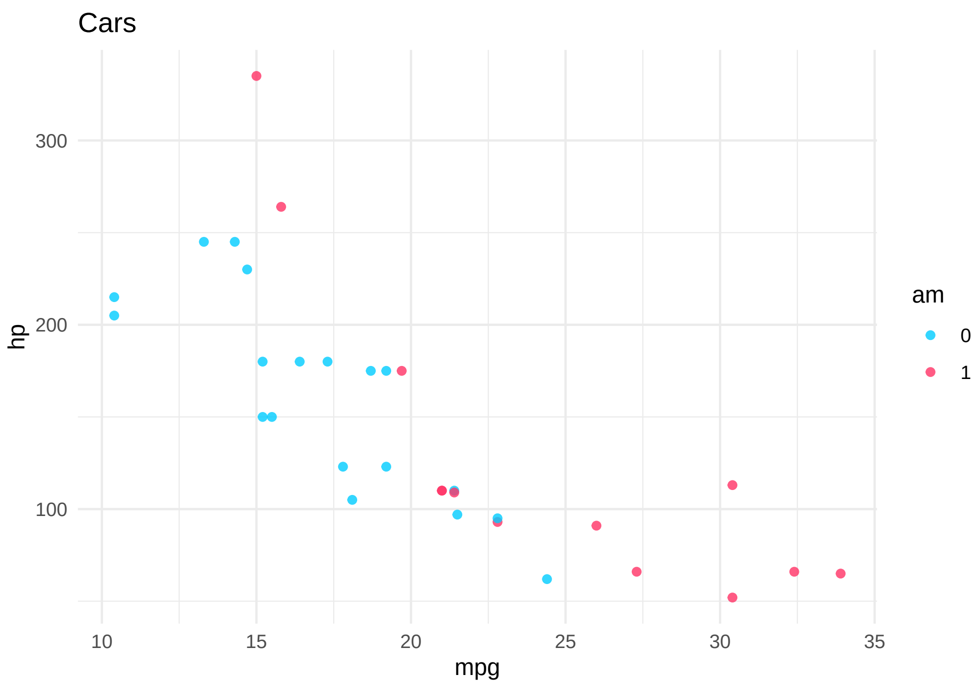
1p



- This plot is for the original data not for the reduced data, however the new direction should be reasonable because PCA rorate the data to along the principle components that explain the most variance.

2)

```
reduced <- cbind(cars,am = mtcars$am)

library(ggplot2)
ggplot(reduced, aes(x = mpg, y = hp))+
  geom_point(aes(color = as.factor(am)),
             size = 1.5,
             alpha = 0.8 )+
  scale_color_manual(values = c('#00CCFF', '#FF3366'))+
  labs(title = "Cars",
       x = "mpg",
       y = "hp",
       colour = "am")+
  theme_minimal()
```



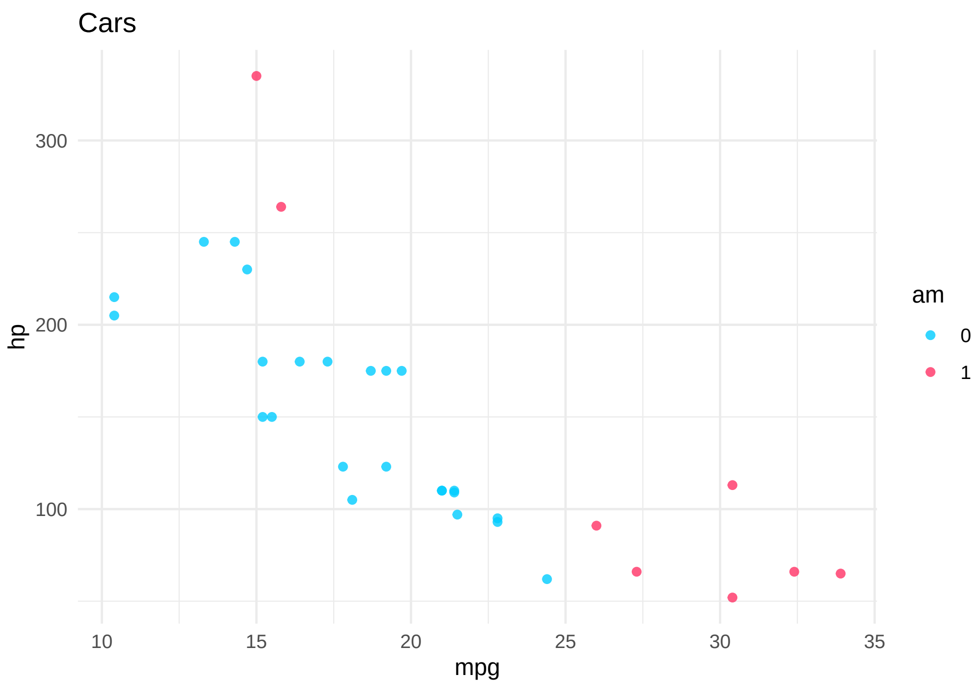

```r
library(MASS)
lda_model <- lda(am ~ mpg + hp, data = reduced)
lda_pred  <- predict(lda_model, data = reduced)

con_mat   <- table("Actuals" = reduced$am, "Predictions" = lda_pred$class)
miss_rate <- 1 - sum(diag(con_mat)) / sum(con_mat)
con_mat
```

```
##         Predictions
## Actuals  0  1
##        0 19  0
##        1  5  8
```

```r
miss_rate
```

```
## [1] 0.15625
```

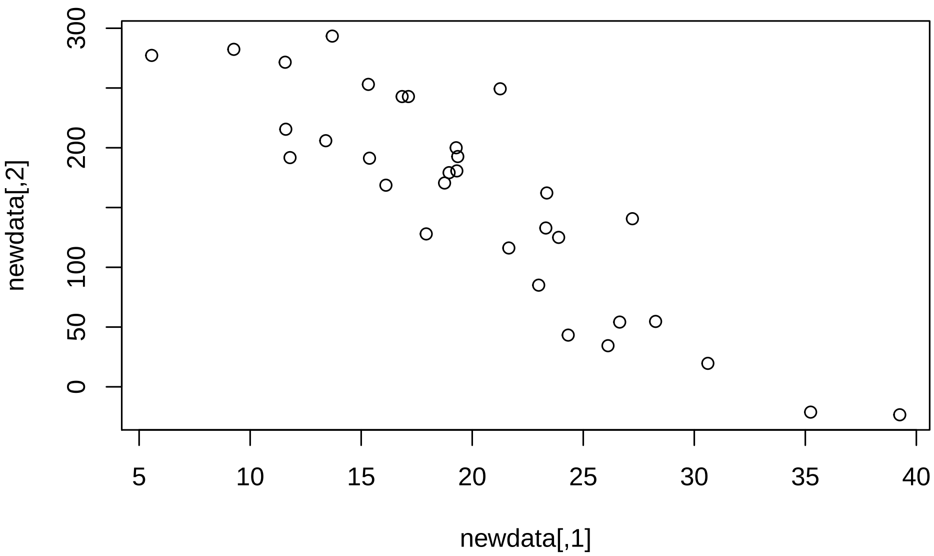
```r
ggplot(reduced, aes(x = mpg, y = hp))+
  geom_point(aes(color = lda_pred$class),
             size = 1.5,
             alpha = 0.8 )+
  scale_color_manual(values = c('#00CCFF', '#FF3366'))+
  labs(title = "Cars",
       x = "mpg",
       y = "hp",
       colour = "am")+
  theme_minimal()
```



1.5p

- Although LDA gave fairly good prediction, the data used violates the lda assumptions about being generated from conditional univariate normal distribution and from equal covariance matrices, which we can see in the plot are violated to some degree, however given that LDA is a robust classification method even if the assumption of normality and common covariance matrix are not satisfied it can in some cases give good prediction.

```
library(mvtnorm)
m1 <- mean(cars$mpg)
m2 <- mean(cars$hp)
n  <- dim(cars)[1]
set.seed(12345)
newdata <- rmvnorm(n, mean = c(m1,m2), sigma = Var1)
plot(newdata)
```



- The simulated data does not look like the original data since we assumed normality for the multivariate distribution that does not exist in the original data

Assignment 3

Ensemble Methods

- Let h(x) denote the true regression. Then, $f^b(x) = h(x) + \epsilon^b(x)$ where b is a boot strab sample
- The Mean Squared Error of $f^b(x)$ can be expressed as:

$$E_x\left[(f^b(x) - h(x))^2\right] = E_x\left[\epsilon^b(x)^2\right]$$

- Therefore we can have Mean Squared Error of $f_{bag}(x)$ can be expressed as:

$$E_x\left[(\frac{1}{B}\sum_b f^b(x) - h(x))^2\right] = E_x\left[\frac{1}{B}\sum_b \epsilon^b(x)^2\right]$$

- When the individual errors have zero mean and are uncorrelated, $E_x[\epsilon^b(x)] = 0$ and $E_x[\epsilon^b(x)\epsilon^{b'}(x)] = 0$ we get

$$E_x\left[(\frac{1}{B}\sum_b f^b(x) - h(x))^2\right] = \frac{1}{B}\left(\frac{1}{B}\sum_b E_x[\epsilon^b(x)^2]\right)$$

```r
ms <- rep(0,10)
s  <- matrix(0,10,10)
for(r in 1:10){
  for(c in 1:10){
    if(r==c){s[r,c] <- 1}
    else{
      s[r,c] <- runif(1,1,2)
    }
    s[c,r] <- s[r,c]
  }
}
B <- 100
b <- rmvnorm(B, mean = ms, sigma = s)
```

Neural Networks

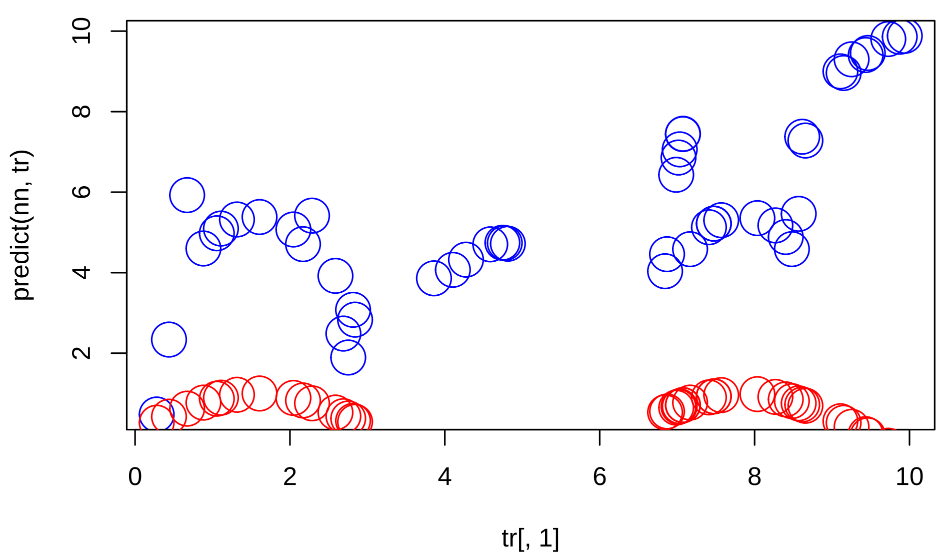
```r
library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 10)
tr  <- data.frame(Var, Sin=sin(Var))
winit <- runif(31, -1, 1)
nn <- neuralnet(formula = Var ~ Sin, data = tr, hidden = 10, startweights = winit, threshold = 0.02, li
```

```
## hidden: 10    thresh: 0.02    rep: 1/1    steps:    1000 min thresh: 0.127831793156119
##                                                     2000 min thresh: 0.0509888320802507
##                                                     3000 min thresh: 0.0509888320802507
##                                                     4000 min thresh: 0.0509888320802507
##                                                     5000 min thresh: 0.0509888320802507
##                                                     6000 min thresh: 0.0509888320802507
##                                                     7000 min thresh: 0.0509888320802507
##                                                     8000 min thresh: 0.0509888320802507
##                                                     9000 min thresh: 0.0509888320802507
##                                                    10000 min thresh: 0.0509888320802507
##                                                    11000 min thresh: 0.0509888320802507
##                                                    12000 min thresh: 0.0509888320802507
##                                                    13000 min thresh: 0.0509888320802507
##                                                    14000 min thresh: 0.0509888320802507
##                                                    15000 min thresh: 0.0509888320802507
##                                                    16000 min thresh: 0.0509888320802507
##                                                    17000 min thresh: 0.0509888320802507
##                                                    18000 min thresh: 0.0509888320802507
##                                                    19000 min thresh: 0.0509888320802507
##                                                    20000 min thresh: 0.0509888320802507
##                                                    21000 min thresh: 0.0509888320802507
##                                                    22000 min thresh: 0.0509888320802507
##                                                    23000 min thresh: 0.0509888320802507
##                                                    24000 min thresh: 0.0509888320802507
##                                                    25000 min thresh: 0.0509888320802507
##                                                    26000 min thresh: 0.0509888320802507
##                                                    27000 min thresh: 0.0509888320802507
##                                                    28000 min thresh: 0.0509888320802507
```

```
##                                                29000 min thresh: 0.0509888320802507
##                                                30000 min thresh: 0.0509888320802507
##                                                31000 min thresh: 0.0509888320802507
##                                                32000 min thresh: 0.0509888320802507
##                                                33000 min thresh: 0.0509888320802507
##                                                34000 min thresh: 0.0509888320802507
##                                                35000 min thresh: 0.0509888320802507
##                                                36000 min thresh: 0.0509888320802507
##                                                37000 min thresh: 0.0509888320802507
##                                                38000 min thresh: 0.0509888320802507
##                                                39000 min thresh: 0.0509888320802507
##                                                40000 min thresh: 0.0509888320802507
##                                                41000 min thresh: 0.0509888320802507
##                                                42000 min thresh: 0.0454262746221461
##                                                43000 min thresh: 0.0454262746221461
##                                                44000 min thresh: 0.0437101134748887
##                                                45000 min thresh: 0.0437101134748887
##                                                46000 min thresh: 0.0383457437434989
##                                                47000 min thresh: 0.0383457437434989
##                                                48000 min thresh: 0.0355273770608871
##                                                49000 min thresh: 0.0345200230341398
##                                                50000 min thresh: 0.0339084887760521
##                                                51000 min thresh: 0.0339084887760521
##                                                52000 min thresh: 0.0339084887760521
##                                                53000 min thresh: 0.0339084887760521
##                                                54000 min thresh: 0.0339084887760521
##                                                55000 min thresh: 0.0339084887760521
##                                                56000 min thresh: 0.0339084887760521
##                                                57000 min thresh: 0.0339084887760521
##                                                58000 min thresh: 0.0339084887760521
##                                                59000 min thresh: 0.0339084887760521
##                                                60000 min thresh: 0.0339084887760521
##                                                61000 min thresh: 0.0339084887760521
##                                                62000 min thresh: 0.0339084887760521
##                                                63000 min thresh: 0.0339084887760521
##                                                64000 min thresh: 0.0339084887760521
##                                                65000 min thresh: 0.0339084887760521
##                                                66000 min thresh: 0.0339084887760521
##                                                67000 min thresh: 0.0339084887760521
##                                                68000 min thresh: 0.0339084887760521
##                                                69000 min thresh: 0.0339084887760521
##                                                70000 min thresh: 0.0339084887760521
##                                                71000 min thresh: 0.0339084887760521
##                                                72000 min thresh: 0.0339084887760521
##                                                73000 min thresh: 0.0339084887760521
##                                                74000 min thresh: 0.0339084887760521
##                                                75000 min thresh: 0.0339084887760521
##                                                76000 min thresh: 0.0339084887760521
##                                                77000 min thresh: 0.0339084887760521
##                                                78000 min thresh: 0.0339084887760521
##                                                79000 min thresh: 0.0339084887760521
##                                                80000 min thresh: 0.0339084887760521
##                                                81000 min thresh: 0.0339084887760521
##                                                82000 min thresh: 0.0339084887760521
```

```
##                                                                83000 min thresh: 0.0339084887760521
##                                                                84000 min thresh: 0.0339084887760521
##                                                                85000 min thresh: 0.0339084887760521
##                                                                86000 min thresh: 0.0339084887760521
##                                                                87000 min thresh: 0.0339084887760521
##                                                                88000 min thresh: 0.0339084887760521
##                                                                89000 min thresh: 0.0339084887760521
##                                                                90000 min thresh: 0.0339084887760521
##                                                                91000 min thresh: 0.0338502968028651
##                                                                92000 min thresh: 0.0322476595312744
##                                                                93000 min thresh: 0.0298116335574625
##                                                                94000 min thresh: 0.0276972206037041
##                                                                95000 min thresh: 0.0234680296525047
##                                                                96000 min thresh: 0.0234680296525047
##                                                                97000 min thresh: 0.0208332627585758
##                                                                98000 min thresh: 0.0200817070944165
##                                                                99000 min thresh: 0.0200817070944165
##                                                                99278 error: 116.84174    time: 18.53 secs
```

```r
plot(tr[,1],predict(nn,tr), col="blue", cex=3)
points(tr, col = "red", cex=3)
```



- In the second case because the variance does not resemble the pattern we get when we predict the sin it just predict chaitic pattern but with the new predicted variance.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE, fig.align = "center", out.width = "80%", warning = FALSE)
RNGversion("3.5.2")
## Assignment 1

temp <- read.csv2("Dailytemperature.csv")
phis <- matrix(0,1792,202)
temp <- cbind(temp,phis)


i <- -50
for(j in 3:103){
  for(obs in 1:1792){
    temp[obs,j]     <- sin(0.5^i * temp$Day[obs])
    temp[obs,j+101] <- cos(0.5^i * temp$Day[obs])
  }
  i <- i + 1
}

hist(temp$Temperature, breaks = 50)
library(glmnet)
lasso <- glmnet(x = as.matrix(temp[,-c(1,2)]),
                y = as.matrix(temp$Temperature),
                alpha = 1,
                family = "gaussian")

plot(lasso, xvar="lambda", label=TRUE)
set.seed(12345)
lasso_cv <- cv.glmnet(x = as.matrix(temp[,-c(1,2)]),
                      y = as.matrix(temp$Temperature),
                      alpha=1,
                      family="gaussian",
                      lambda = 0:100 * 0.001)

plot(lasso_cv)

c("Minimum Lambda" = lasso_cv$lambda.min)
c("1se Lambda" = lasso_cv$lambda.1se)

lasso_opt <- glmnet(x = as.matrix(temp[,-c(1,2)]),
                  y = as.matrix(temp$Temperature),
                  alpha = 1,
                  family = "gaussian",
                  lambda = lasso_cv$lambda.min)
c("Number of non-zero Features" = lasso_opt$df)

coef(lasso_opt, s = lasso_cv$lambda.min)
pred <- predict(lasso_opt, newx = as.matrix(temp[,-c(1,2)]))

mydata <- data.frame(x = temp$Day, y = temp$Temperature, yhat = pred)

plot(temp$Temperature, type = "b", col = "red")
points(pred, type = "b", col = "blue")
data(mtcars)
```

```r
cars <- mtcars[,c(1,4)]
Var1  <- var(cars)

comps <- eigen(Var1)$vectors
colnames(comps) <- c("PC1", "PC2")

c("First Component" = comps[,1])
plot(cars, main = "Original Data")
reduced <- cbind(cars,am = mtcars$am)

library(ggplot2)
ggplot(reduced, aes(x = mpg, y = hp))+
  geom_point(aes(color = as.factor(am)),
             size = 1.5,
             alpha = 0.8 )+
  scale_color_manual(values = c('#00CCFF', '#FF3366'))+
  labs(title = "Cars",
       x = "mpg",
       y = "hp",
       colour = "am")+
  theme_minimal()

library(MASS)
lda_model <- lda(am ~ mpg + hp, data = reduced)
lda_pred  <- predict(lda_model, data = reduced)

con_mat   <- table("Actuals" = reduced$am, "Predictions" = lda_pred$class)
miss_rate <- 1 - sum(diag(con_mat)) / sum(con_mat)
con_mat
miss_rate

ggplot(reduced, aes(x = mpg, y = hp))+
  geom_point(aes(color = lda_pred$class),
             size = 1.5,
             alpha = 0.8 )+
  scale_color_manual(values = c('#00CCFF', '#FF3366'))+
  labs(title = "Cars",
       x = "mpg",
       y = "hp",
       colour = "am")+
  theme_minimal()
library(mvtnorm)
m1 <- mean(cars$mpg)
m2 <- mean(cars$hp)
n  <- dim(cars)[1]
set.seed(12345)
newdata <- rmvnorm(n, mean = c(m1,m2), sigma = Var1)
plot(newdata)
ms <- rep(0,10)
s  <- matrix(0,10,10)
for(r in 1:10){
  for(c in 1:10){
    if(r==c){s[r,c] <- 1}
```

```
    else{
      s[r,c] <- runif(1,1,2)
    }
    s[c,r] <- s[r,c]
  }
}
B <- 100
b <- rmvnorm(B, mean = ms, sigma = s)
library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 10)
tr  <- data.frame(Var, Sin=sin(Var))
winit <- runif(31, -1, 1)
nn <- neuralnet(formula = Var ~ Sin, data = tr, hidden = 10, startweights = winit, threshold = 0.02, li
plot(tr[,1],predict(nn,tr), col="blue", cex=3)
points(tr, col = "red", cex=3)
```