Homework 4 Yan Liu

Problem 1

(c)

```r
library(dplyr, quietly = T)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(magrittr)
# The function below is adapted from the HW3 solution
exp_logL <- function(X, theta)
{
  p1 <- theta["p1"]; p2 <- 1-p1
  mu1 <- theta["mu1"]; mu2 <- theta["mu2"]
  s1 <- theta["sigma1"]; s2 <- theta["sigma2"]

  N1 <- dnorm(X, mean=mu1, sd=sqrt(s1))
  N2 <- dnorm(X, mean=mu2, sd=sqrt(s2))
  D <- p1*N1 + p2*N2

  r1 <- p1*N1/D
  r2 <- 1-r1

  exp_ll <- (r1*log(p1*N1) + r2*log(p2*N2)) %>% sum
  return (exp_ll)
}

# test
theta <- c(mu1=70, mu2=60, sigma1=10, sigma2=15, p1=.3)
X <- read.table("Hope Heights.txt", header=T) %>% dplyr::pull(Height)

print(exp_logL(X, theta))
```

```
## [1] -378.3055
```

```r
grad_logL <- function(X, theta)
{
  p1 <- theta["p1"]; p2 <- 1-p1
  mu1 <- theta["mu1"]; mu2 <- theta["mu2"]
  s1 <- theta["sigma1"]; s2 <- theta["sigma2"]

  N1 <- dnorm(X, mean=mu1, sd=sqrt(s1))
```

```r
  N2 <- dnorm(X, mean=mu2, sd=sqrt(s2))

  D <- p1*N1 + p2*N2
  r1 <- p1*N1/D
  r2 <- 1-r1
  r1.sum <- r1 %>% sum
  r2.sum <- r2 %>% sum


  N <- length(X)
  grad_samples <- matrix(c(mu1=r1*X/r1.sum,
                           mu2=r2*X/r2.sum,
                           s1 = r1*(X-mu1)^2/r1.sum,
                           s2 = r2*(X-mu2)^2/r2.sum,
                           p1= r1.sum/N),
                         nrow=N,
                         ncol=length(theta))

  grad <- grad_samples %>% colSums %>% setNames(c("mu1", "mu2", "s1", "s2", "p1"))
  return (grad_samples %>% colSums)
}

# test
print(grad_logL(X, theta))
```

```
## Warning in matrix(c(mu1 = r1 * X/r1.sum, mu2 = r2 * X/r2.sum, s1 = r1 * :
## data length [401] is not a sub-multiple or multiple of the number of rows
## [100]

## [1] 71.01751 64.84814 10.88781 31.57688 70.80153
```

```r
norm <- function(x) sqrt(sum(x^2))

expectation_maximization <- function(X, start_theta,
                                     max_iter=100)
{
  theta <- start_theta
  iter <- 1

  g <- grad_logL(X, theta)
  while (norm(g) > 1E-3 & iter <= max_iter) {

    if (iter %% 10 == 0)
    cat("iter=", iter, "likelihood=", exp_logL(X, theta), "\n")

    ng <- g/norm(g)
    s <- 1
    new_theta <- theta + s*ng

    # back up if we exceed boundaries
    while(new_theta["p1"] < 0 | new_theta["p1"] > 1 |
          new_theta["sigma1"] < 0 | new_theta["sigma2"] < 0) {
      s <- s/2
      new_theta <- theta + s*ng
    }
```

```r
    # backtrack
    while(exp_logL(X, new_theta) < exp_logL(X, theta)) {
      s <- s/2
      new_theta <- theta + s*ng
    }

    theta <- new_theta
    g <- grad_logL(X, theta)
    iter <- iter + 1
  }

  return (list(theta=theta,
               likelihood=exp_logL(X, theta)))
}

# try a few starting points
start_theta <- c(mu1=75, mu2=65, sigma1=5, sigma2=5, p1=.3)
out1 <- expectation_maximization(X, start_theta, max_iter=100)
```

```
## iter= 10 likelihood= -327.9474
## iter= 20 likelihood= -327.9474
## iter= 30 likelihood= -327.9474
## iter= 40 likelihood= -327.9474
## iter= 50 likelihood= -327.9474
## iter= 60 likelihood= -327.9474
## iter= 70 likelihood= -327.9474
## iter= 80 likelihood= -327.9474
## iter= 90 likelihood= -327.9474
## iter= 100 likelihood= -327.9474
```

```r
print(out1)
```

```
## $theta
##        mu1        mu2     sigma1     sigma2         p1
## 75.1488221 65.1353336  5.0138827  5.0206773  0.4463439
##
## $likelihood
## [1] -327.9474
```

```r
start_theta <- c(mu1=10, mu2=20, sigma1=20, sigma2=20, p1=.5)
out2 <- expectation_maximization(X, start_theta, max_iter=100)
```

```
## iter= 10 likelihood= -5076.949
## iter= 20 likelihood= -4208.267
## iter= 30 likelihood= -3823.811
## iter= 40 likelihood= -3823.811
## iter= 50 likelihood= -3823.811
## iter= 60 likelihood= -3823.811
## iter= 70 likelihood= -3823.811
## iter= 80 likelihood= -3823.811
## iter= 90 likelihood= -3823.811
## iter= 100 likelihood= -3823.811
```

```r
print(out2)
```

```
## $theta
##        mu1        mu2     sigma1     sigma2         p1
## 10.4694963 20.5010002 41.8108540 37.5706493  0.9678372
##
## $likelihood
## [1] -3823.811
```

```r
start_theta <- c(mu1=80, mu2=60, sigma1=5, sigma2=5, p1=.2)
out3 <- expectation_maximization(X, start_theta, max_iter=100)
```

```
## iter= 10 likelihood= -721.0258
## iter= 20 likelihood= -721.0258
## iter= 30 likelihood= -721.0258
## iter= 40 likelihood= -721.0258
## iter= 50 likelihood= -721.0258
## iter= 60 likelihood= -721.0258
## iter= 70 likelihood= -721.0258
## iter= 80 likelihood= -721.0258
## iter= 90 likelihood= -721.0258
## iter= 100 likelihood= -721.0258
```

```r
print(out3)
```

```
## $theta
##        mu1        mu2     sigma1     sigma2         p1
## 80.5202222 60.4723872  5.3427470  5.3570546  0.7111318
##
## $likelihood
## [1] -721.0258
```

The first starting point gives the highest likelihood and seems to split the heights roughly as we would expect (although only because we know the answer).

(d)

Let's choose the mixture based on which of $P(z_i = 1 \mid \hat{X}_i, \theta)$ and $P(z_i = 2 \mid \hat{X}_i, \theta)$ is bigger.

```r
# Get the theta that was best from b.i
theta <- out1$theta
theta
```

```
##        mu1        mu2     sigma1     sigma2         p1
## 75.1488221 65.1353336  5.0138827  5.0206773  0.4463439
```

```r
Gender <- read.table("Hope Heights.txt", header=T) %>% dplyr::pull(Gender)

classify_gender <- function(X, theta)
{
  p1 <- theta["p1"]; p2 <- 1-p1
  mu1 <- theta["mu1"]; mu2 <- theta["mu2"]
  s1 <- theta["sigma1"]; s2 <- theta["sigma2"]

  N1 <- dnorm(X, mean=mu1, sd=sqrt(s1))
  N2 <- dnorm(X, mean=mu2, sd=sqrt(s2))

  D <- p1*N1 + p2*N2
```

```r
  r1 <- p1*N1/D
  r2 <- 1-r1

  # The first mixture has higher mu, so associate that with male gender
  guess <- ifelse(r1*log(p1*N1) > r2*log(p2*N2), 1, 2)

  return (guess)
}

guess_Gender <- classify_gender(X, theta)
correct <- sum(Gender==guess_Gender)/length(Gender)
print(correct)
```

```
## [1] 0.85
```

We get roughly 85% right, which is better than in Homework 3. Medium heights is harder to classify or guess that extream heights.

```r
data.frame(height=X, guess=guess_Gender, true=Gender) %>% dplyr::arrange(height) -> result
result %>% head(5)
```

```
##   height guess true
## 1     60     1    1
## 2     60     1    1
## 3     61     1    1
## 4     61     1    1
## 5     61     1    1
```

```r
result[45:55,]
```

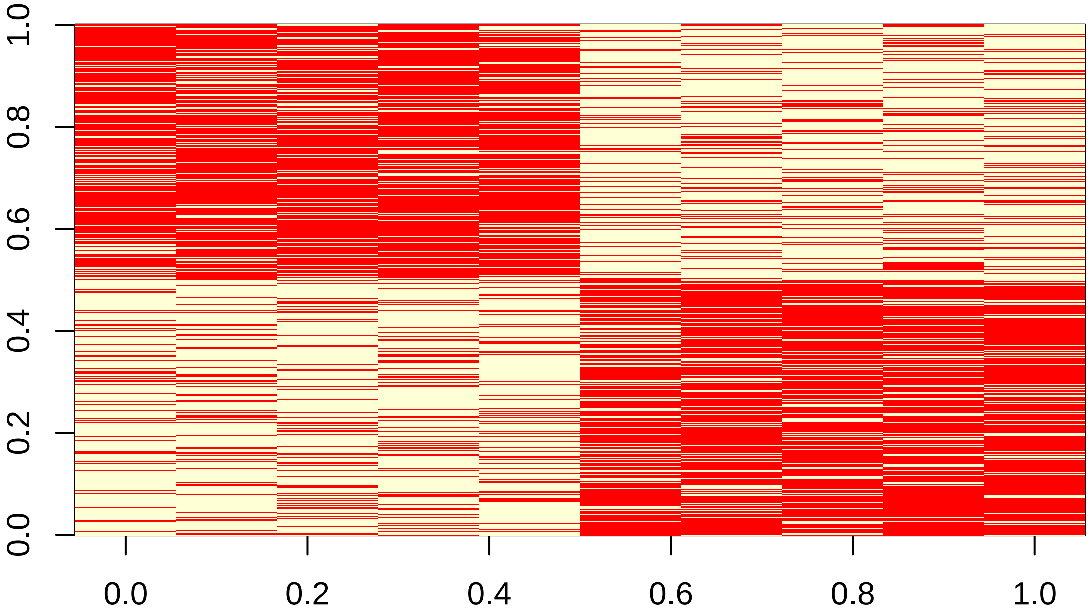
```
##    height guess true
## 45     69     1    1
## 46     69     1    1
## 47     69     1    1
## 48     69     1    2
## 49     69     1    2
## 50     69     1    2
## 51     69     1    2
## 52     70     1    1
## 53     70     1    1
## 54     70     1    1
## 55     70     1    2
```

```r
result %>% tail(5)
```

```
##     height guess true
## 96      75     2    2
## 97      76     2    2
## 98      76     2    2
## 99      77     2    2
## 100     79     2    2
```

Problem 2

```r
X_b <- read.csv("noisy_bits.csv")
image(t(X_b))
```




```r
exp_b_logL <- function(X_b, theta_b)
{
  p1 <- theta_b["p1",]; p2 <- 1-p1
  mu1 <- theta_b["mu1",]; mu2 <- theta_b["mu2",]

  # mu1=.2;
  # mu2=.2; p1=.45;
  # p2 <- 1-p1

  R1 <- p1*apply(mu1^X_b*(1-mu1)^(1-X_b), 1, prod)
  R2 <- p2*apply(mu2^X_b*(1-mu2)^(1-X_b), 1, prod)
  r1 <- R1 / (R1+R2)
  r2 <- 1-r1

  exp_b_ll <- (r1*log(R1) + r2*log(R2)) %>% sum
  return (exp_b_ll)
}

# test
theta_b<- matrix(rep(c(.4, .2, p1=.45), each =10), nrow =3, byrow = T) %>%
            set_rownames(c("mu1", "mu2", "p1"))
theta_b
```

```
##     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## mu1 0.40 0.40 0.40 0.40 0.40 0.40 0.40 0.40 0.40  0.40
## mu2 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20  0.20
## p1  0.45 0.45 0.45 0.45 0.45 0.45 0.45 0.45 0.45  0.45
```

```r
#theta_b
print(exp_b_logL(X_b, theta_b))
```

```
## [1] -4036.815
```

```r
grad_b_logL <- function(X_b, theta_b)
{
  p1 <- theta_b["p1",]; p2 <- 1-p1
  mu1 <- theta_b["mu1",]; mu2 <- theta_b["mu2",]

  R1 <- p1*apply(mu1^X_b*(1-mu1)^(1-X_b), 1, prod)
  R2 <- p2*apply(mu2^X_b*(1-mu2)^(1-X_b), 1, prod)
  r1 <- as.vector(R1 / (R1+R2))
  r2 <- 1-r1

  r1.sum <- r1 %>% sum
  r2.sum <- r2 %>% sum


  N <- dim(X_b)[1]
  dimension <- dim(X_b)[2]
  grad_samples <- matrix(c(mu1=(t(r1) %*% as.matrix(X_b)),
                           mu2=(t(r2) %*% as.matrix(X_b)),
                           p1= rep(r1.sum/N, dimension)),
                           byrow = T,
                           ncol=dimension) %>% set_rownames(c("mu1", "mu2", "p1"))
  return (grad_samples)
}

# test
print(grad_b_logL(X_b, theta_b))
```

```
##             [,1]        [,2]        [,3]        [,4]        [,5]
## mu1 221.1760721 215.2652768 211.0252129 198.2710868 218.7350830
## mu2  38.8239279  37.7347232  39.9747871  35.7289132  36.2649170
## p1    0.8040731   0.8040731   0.8040731   0.8040731   0.8040731
##             [,6]        [,7]        [,8]        [,9]       [,10]
## mu1 228.2423173 211.1050655 214.3007194 205.4398974 209.2377088
## mu2  43.7576827  36.8949345  38.6992806  33.5601026  33.7622912
## p1    0.8040731   0.8040731   0.8040731   0.8040731   0.8040731
```

```r
expectation_maximization_b <- function(X_b, start_theta_b,
                                max_iter=200)
{
  theta_b <- start_theta_b
  iter <- 1
  s <- 1
  gradient <- grad_b_logL(X_b, theta_b)

  while (iter <= max_iter) {

    # if (iter %% 10 == 0)
    #    cat("iter=", iter, "likelihood=", exp_b_logL(X_b, theta_b), "\n")

    ng <- gradient/norm(gradient)
```

```r
    s <- 1
    new_theta_b <- theta_b + s*ng

    # back up if we exceed boundaries
    while(any(new_theta_b["p1",]) < 0) {
      s <- s/2
      new_theta_b <- theta_b + s*ng
    }

     # backtrack

    while((exp_b_logL(X_b, new_theta_b) < exp_b_logL(X_b, theta_b))) {
       s <- s/2
       new_theta_b <- theta_b + s*ng
    }

    theta_b <- new_theta_b
    gradient <- grad_b_logL(X_b, theta_b)
    iter <- iter + 1

    return (list(theta=theta_b,
             likelihood=exp_b_logL(X_b, theta_b)))
  }
}


#start_theta <- c(mu1=.4, mu2=.3,  p1=.4)
start_theta_b<- matrix(rep(c(.4, .2, p1=.2), each =10), nrow =3, byrow = T) %>%
                  set_rownames(c("mu1", "mu2", "p1"))
final_out <- expectation_maximization_b(X_b, start_theta_b, max_iter=200)
```

```r
# Get the theta that was best from b.i
theta <- final_out$theta

classify_X <- function(X, theta)
{
  p1 <- theta["p1",];
  p2 <- 1-p1;
  mu1 <- theta["mu1",];
  mu2 <- theta["mu2",]

  R1 <- p1*apply(mu1^X_b*(1-mu1)^(1-X_b), 1, prod)
  R2 <- p2*apply(mu2^X_b*(1-mu2)^(1-X_b), 1, prod)
  r1 <- R1 / (R1+R2)
  r2 <- 1-r1

  guess <- ifelse((r1*log(R1) > r2*log(R2)), 0, 1)

  return (guess)
}

guess_X <- classify_X(X_b, theta)
X_b_label <- ifelse(apply(X_b, 1, sum)>5,1,0)
```

```r
print(sum(guess_X == X_b_label)/500)
```

```
## [1] 0.87
```

The result is fairly good.