STAT34800 HW4

Sarah Adilijiang

Problem A

(1) EM update function

EM algorighm:

1. Initialize $\theta^0 = \{\mu_k's, \sigma_k's, \pi_k's\}$, and evaluate the incomplete data log-likelihood with these parameters:

$$l(\theta^0) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \ N(x_i; \mu_k, \sigma_k^2) \right)$$

2. **E-step**: Evaluate the posterior probabilities $\gamma_{Z_i}(k)$ using the current values of the μ_k, σ_k^2, π_k:

$$\gamma_{Z_i}(k) = P(Z_i = k|X_i) = \frac{P(Z_i = k)P(X_i|Z_i = k)}{P(X_i)} = \frac{\pi_k \ N(x_i; \mu_k, \sigma_k^2)}{\sum_{k=1}^{K} \pi_k \ N(x_i; \mu_k, \sigma_k^2)}$$

3. **M-step**: use the current values of $\gamma_{Z_i}(k)$ to find the expectation of the complete data log-likelihood $Q(\theta, \theta^0)$, evaluated at an arbitrary θ:

$$Q(\theta, \theta^0) = E_{Z|X,\theta^0} \left[\log(P(X, Z|\theta))\right] = E_{Z|X,\theta^0} \left[\sum_{i=1}^{n} \sum_{k=1}^{K} I(Z_i = k) \left(\log(\pi_k) + \log(N(x_i; \mu_k, \sigma_k^2))\right) \right]$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{Z_i}(k) \left(\log(\pi_k) + \log(N(x_i; \mu_k, \sigma_k^2))\right) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{Z_i}(k) \left(\log(\pi_k) - \frac{1}{2} \log(2\pi\sigma_k^2) - \frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right)$$

Then by setting derivatives to zeros:

$$\frac{\partial Q(\theta, \theta^0)}{\partial \mu_k} = 0, \quad \frac{\partial Q(\theta, \theta^0)}{\partial \sigma_k^2} = 0, \quad \frac{\partial Q(\theta, \theta^0)}{\partial \pi_k} = 0$$

We can get the MLE estimates of new parameters $\hat{\mu}_k, \hat{\sigma}_k^2, \hat{\pi}_k$ with the current values of $\gamma_{Z_i}(k)$ that maximizes the complete data log-likelihood $Q(\theta, \theta^0)$, i.e. $\hat{\theta} = argmax_\theta \ Q(\theta, \theta^0)$:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma_{Z_i}(k) \ x_i$$

$$\hat{\sigma}_k^2 = \frac{1}{N_k} \sum_{i=1}^{n} \gamma_{Z_i}(k) \ (x_i - \mu_k)^2$$

$$\hat{\pi}_k = \frac{N_k}{n}$$

where

$$N_k = \sum_{i=1}^{n} \gamma_{Z_i}(k)$$

4. Evaluate the incomplete data log-likelihood with the new parameter estimates:

$$l(\hat{\theta}) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \hat{\pi}_k \ N(x_i; \hat{\mu}_k, \hat{\sigma}_k^2) \right)$$

If the log-likelihood has changed by less than some small ϵ, stop. Otherwise, go back to E-step.

EM function

```
# compute the incomplete data log-likelihood
compute.log.lik = function(X, L, w) {
    for (i in 1:ncol(L)) {
        L[,i] = L[,i]*w[i]
    }
    return(sum(log(rowSums(L))))
}


# EM algorithm function
mixture.EM = function(X, K, w.init, mu.init, sigma.init) {

    # initialize parameters
    w.curr = w.init
    mu.curr = mu.init
    sigma.curr = sigma.init

    # compute the likelihood P(Xi|Zi=k) = N(Xi; mu, sigma)
    L = matrix(NA, nrow=length(X), ncol=K)
    for (i in 1:K) {
        L[,i] = dnorm(X, mean=mu.curr[i], sd=sigma.curr[i])
    }

    # compute & store incomplete data log-likehoods for each iteration
    log_liks = c()
    ll       = compute.log.lik(X, L, w.curr)
    log_liks = c(log_liks, ll)


    # EM steps & checks for convergence
    delta.ll = 1
    while(delta.ll > 1e-5) {

        # E-step: compute E_{Z|X,w0}[I(Z_i = k)]
        z_ik = L
        for(i in 1:K) {
            z_ik[,i] = w.curr[i]*z_ik[,i]
        }
        z_ik = z_ik / rowSums(z_ik)


        # M-step: update estimates
        N_k = colSums(z_ik)
        w.curr     = N_k/length(X)
        mu.curr    = as.vector( t(z_ik) %*% X / N_k )

        sq = matrix(NA, length(X), K)
        for (i in 1:K) {
            sq[,i] = (X-mu.curr[i])^2
        }
        sigma.curr = sqrt( diag(t(z_ik) %*% sq) / N_k )
```

```
        # update likelihood P(Xi|Zi=k) = N(Xi; mu, sigma)
        L = matrix(NA, nrow=length(X), ncol=K)
        for (i in 1:K) {
            L[,i] = dnorm(X, mean=mu.curr[i], sd=sigma.curr[i])
        }

        # checks for convergence via incomplete data log-likelihoods at each step
        ll       = compute.log.lik(X, L, w.curr)
        log_liks = c(log_liks, ll)
        delta.ll = log_liks[length(log_liks)]  - log_liks[length(log_liks)-1]
    }

    return(list(w=w.curr, mu=mu.curr, sigma=sigma.curr, Z=z_ik, log.liks=log_liks))
}
```

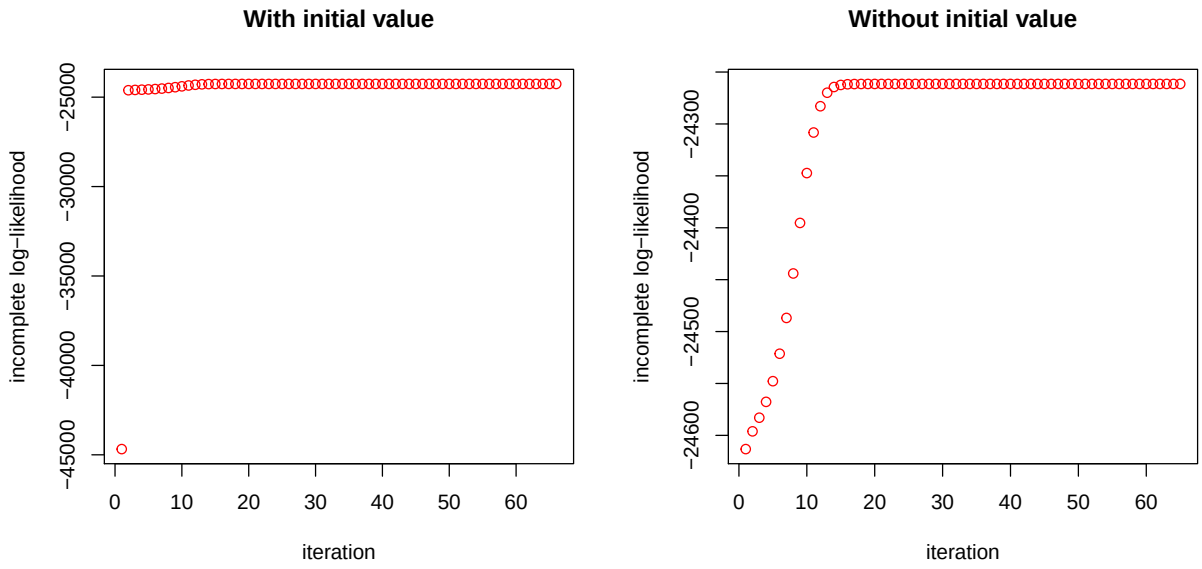
(2) Demonstration by simulation

```
# simulate data
set.seed(123)

# mixture components
mu.true    = c(5, 10)
sigma.true = c(1.5, 2)

# determine Z_i
w.true = c(0.25, 0.75)
Z.true = rbinom(500, 1, 0.75)

# sample data from mixture model
X = rnorm(10000, mean=mu.true[Z.true+1], sd=sigma.true[Z.true+1])
```

(i) log-likelihood strictly increases

```
# perform EM
results2 = mixture.EM(X, K=2, w.init=c(0.5,0.5), mu.init=c(1,2), sigma.init=c(3,4))
```

```
# inspect the evolution of the incomplete log-likelihood
par(mfrow=c(1,2))
plot(results2$log.liks, col=2, ylab='incomplete log-likelihood', xlab='iteration',
     main="With initial value")
plot(results2$log.liks[-1], col=2, ylab='incomplete log-likelihood', xlab='iteration',
     main="Without initial value")   # removing the initial value
```



(ii) final estimates vs true values

```
# compare estimates with true values
compare = round(rbind(w.true, results2$w), 3)
rownames(compare)[2]="w.estimate"; compare
```

```
##             [,1]  [,2]
## w.true     0.250 0.750
## w.estimate 0.241 0.759
```

```
compare = round(rbind(mu.true, results2$mu), 3)
rownames(compare)[2]="mu.estimate"; compare
```

```
##              [,1]   [,2]
## mu.true     5.000 10.000
## mu.estimate 5.014 10.015
```

```
compare = round(rbind(sigma.true, results2$sigma), 3)
rownames(compare)[2]="sigma.estimate"; compare
```

```
##                 [,1] [,2]
## sigma.true     1.500 2.00
## sigma.estimate 1.475 1.99
```

Comments:

(1) In the plots, we see that the incomplete data log-likelihood strictly increases every iteration.

(2) The final results show that the final estimated parameters by EM algorithm are very close to the true values used in the data simulation.

(3) Initial values & local optima

K=3

```
# perform EM
results3 = mixture.EM(X, K=3, w.init=c(0.1,0.3,0.6), mu.init=c(0,1,2), sigma.init=c(3,2,1))
```

```r
# compare estimates with true values
compare = round(rbind(c(w.true,NA), results3$w), 3)
rownames(compare)=c("w.true","w.estimate"); compare
```

```
##             [,1]  [,2]  [,3]
## w.true     0.250 0.750    NA
## w.estimate 0.759 0.164 0.077
```

```r
compare = round(rbind(c(mu.true,NA), results3$mu), 3)
rownames(compare)=c("mu.true","mu.estimate"); compare
```

```
##               [,1]  [,2]  [,3]
## mu.true      5.000 10.00    NA
## mu.estimate 10.014  5.47 4.006
```

```r
compare = round(rbind(c(sigma.true,NA), results3$sigma), 3)
rownames(compare)=c("sigma.true","sigma.estimate"); compare
```

```
##                 [,1]  [,2]  [,3]
## sigma.true     1.500 2.000    NA
## sigma.estimate 1.989 1.313 1.244
```

K=4

```r
# perform EM
results4 = mixture.EM(X, K=4, w.init=c(0.1,0.2,0.3,0.4), mu.init=c(0,2,4,6), sigma.init=c(4,3,2,1))

# compare estimates with true values
compare = round(rbind(c(w.true,NA,NA), results4$w), 3)
rownames(compare)=c("w.true","w.estimate"); compare
```

```
##             [,1]  [,2] [,3]  [,4]
## w.true     0.250 0.750   NA    NA
## w.estimate 0.259 0.303 0.23 0.208
```

```r
compare = round(rbind(c(mu.true,NA,NA), results4$mu), 3)
rownames(compare)=c("mu.true","mu.estimate"); compare
```

```
##               [,1]   [,2] [,3]  [,4]
## mu.true      5.000 10.000   NA    NA
## mu.estimate 10.218 10.692 8.29 4.877
```

```r
compare = round(rbind(c(sigma.true,NA,NA), results4$sigma), 3)
rownames(compare)=c("sigma.true","sigma.estimate"); compare
```

```
##                 [,1]  [,2]  [,3]  [,4]
## sigma.true      1.50 2.000    NA    NA
## sigma.estimate 1.57 2.003 2.045 1.435
```

incomplete log-likelihood

```r
# compare incomplete log-likelihood
results2$log.liks[length(results2$log.liks)]  # K=2
```

```
## [1] -24261.46
```

```r
results3$log.liks[length(results3$log.liks)]  # K=3
```

```
## [1] -24261.16
```

```
results4$log.liks[length(results4$log.liks)]  # K=4
```

```
## [1] -24260.25
```

Comments:

I used two sets of different intial values:

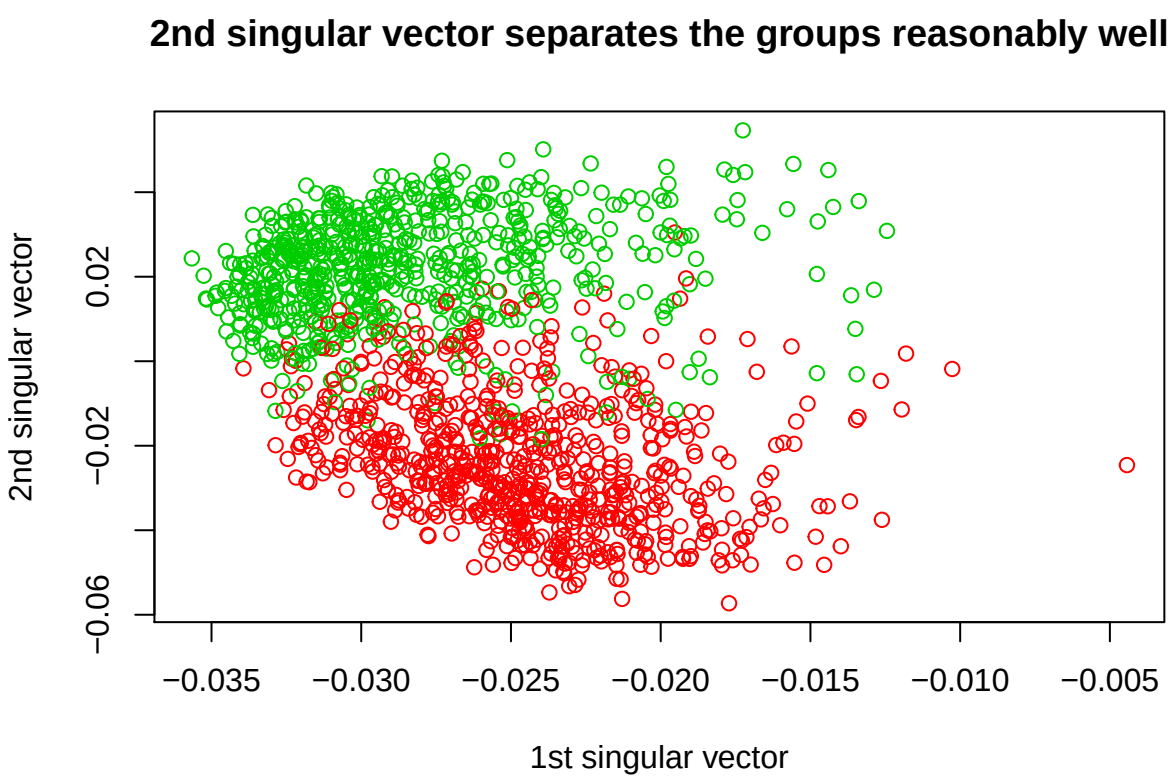
(a) $K = 3, \pi = (0.1, 0.3, 0.6), \mu = (0, 1, 2), \sigma = (1, 1, 1)$

(b) $K = 4, \pi = (0.1, 0.2, 0.3, 0.4), \mu = (0, 2, 4, 6), \sigma = (4, 3, 2, 1)$

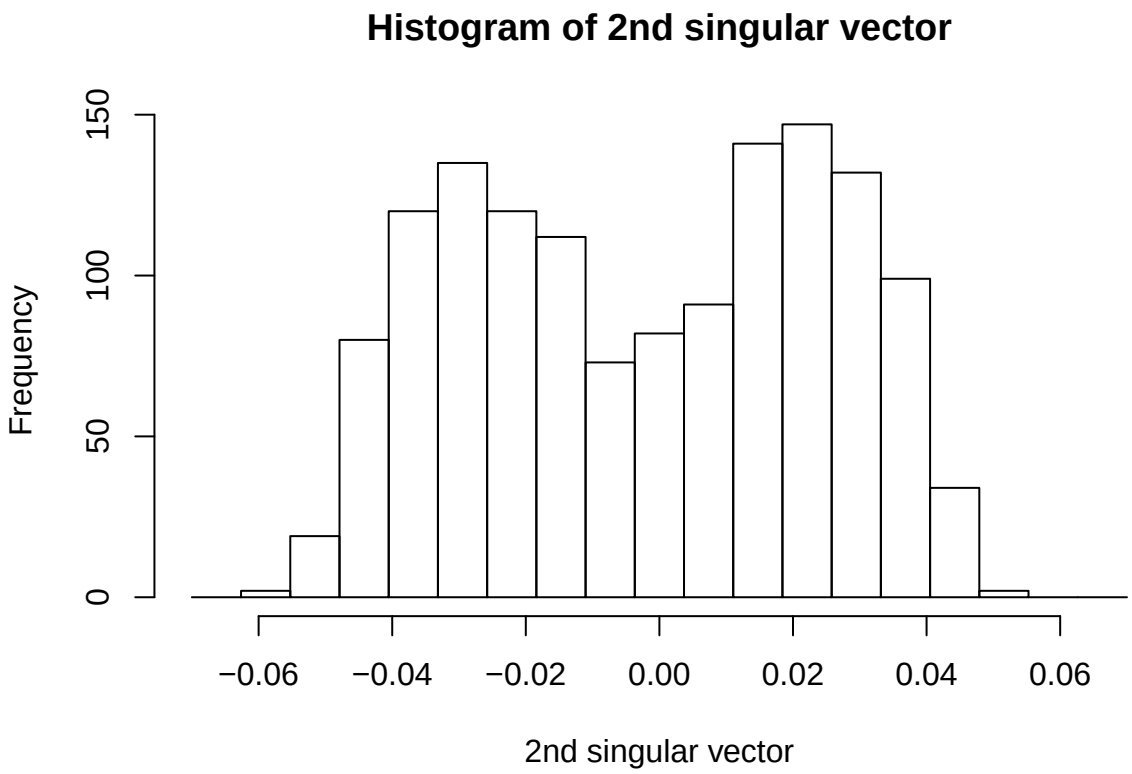
The results show that the estimated values of parameters are different with the true values. And the incomplete log-likelihood of these two experiments are different with each other, also different with the case in question (2) where $K = 2$, though being very close. These results demonstrate that the EM hill-climbing algorithm often get stuck in local optima, so the final solution can depend on the initial values used in the EM algorithm.

(4) Zipcode data

```
# read date & subset 2's and 3's
z = read.table("zip.train.txt")
sub = (z[,1]==2) | (z[,1]==3)
z23 = as.matrix(z[sub, ])

# perform SVD
z23.svd = svd(z23[ ,-1])  # first column are labels

# plot the first two two singular vectors
plot(z23.svd$u[,1], z23.svd$u[,2], col=z23[,1],
     xlab="1st singular vector", ylab="2nd singular vector", main="2nd singular vector separates the gr
```




```
# histogram of the 2nd singular vector
hist(z23.svd$u[,2], breaks=seq(-0.07,0.07,length=20), xlab="2nd singular vector",
     main="Histogram of 2nd singular vector")
```



The histogram suggests a mixture of two Gaussians might be a reasonable start for the 2nd singular vector.

```
# perform EM for a mixture of two Gaussians for the 2nd singular vector
# use multiple initial values within the range of X
X = z23.svd$u[,2]
results = list()
log_Liks = rep(NA,100)

for (i in 1:100) {
    # randomly initial values
    w = runif(1,0,1)
    w = c(w, 1-w)
    mu = runif(2, -0.07, 0.07)  # within the range of X
    sigma = abs(rnorm(2,0,1))

    # EM algorithm
    results[[i]] = mixture.EM(X, K=2, w.init=w, mu.init=mu, sigma.init=sigma)

    # store incomplete log-likelihood
    log_Liks[i] = results[[i]]$log.liks[length(results[[i]]$log.liks)]
}
```

```
# select the solution with the highest log-likelihood
index = which.max(log_Liks)
final.result = results[[index]]
w = final.result$w;          round(w,4)
```

```
## [1] 0.4879 0.5121
```

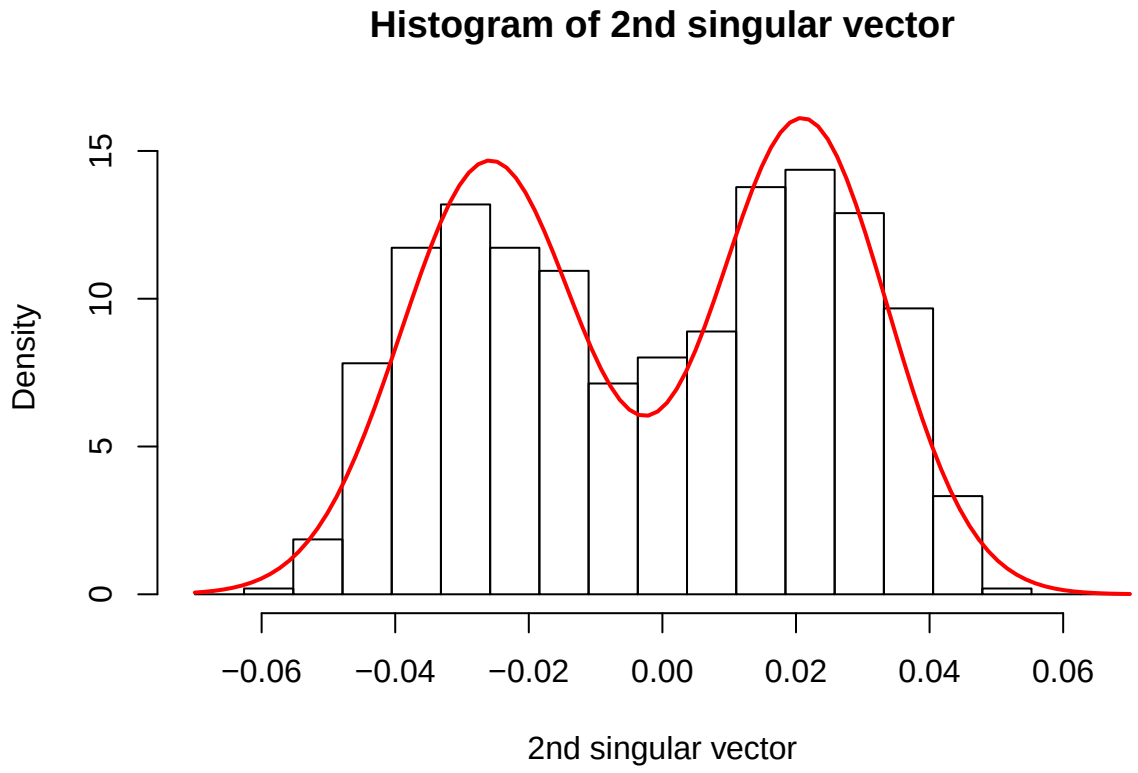
```
mu = final.result$mu;         round(mu,4)
```

```
## [1] -0.0259  0.0210
```

```
sigma = final.result$sigma;  round(sigma,4)
```

```
## [1] 0.0133 0.0127
```

```
# plot fitted mixture density with the histogram
x = seq(-0.07,0.07,length=100)
density = w[1] * dnorm(x, mean=mu[1], sd=sigma[1]) +
          w[2] * dnorm(x, mean=mu[2], sd=sigma[2])
hist(X, breaks=seq(-0.07,0.07,length=20), probability=TRUE, ylim=c(0,16),
     xlab="2nd singular vector", main="Histogram of 2nd singular vector")
lines(density~x, col=2, lwd=2)
```




```
# classification error rate
index = apply(final.result$Z, 1, which.max)   # 1 or 2
class = index + 1     # 2 or 3
label = z23[,1]
mis_rate = min(sum(class!=label), length(class)-sum(class!=label)) / length(class)
mis_rate
```

```
## [1] 0.08207343
```

Results:

(1) The best fit with highest log-likelihood gives the final estimated parameters:

$$\hat{\pi} = c(0.4879, 0.5121), \quad \hat{\mu} = c(-0.0259, 0.0210), \quad \hat{\sigma} = c(0.0133, 0.0127)$$

(2) The fitted mixture density fits the data reasonably well.

(3) The misclassification rate is about 0.08207343, which is low, so our fit is well.