```r
# Load data
setwd("C:/Users/irvin/OneDrive/UChicago/2016-2017 Academic Year/Spring 2017/STAT 24610/Project/Data")
load('digits.RData')
library(expm)
library(Matrix)

image(t(1 - training.data[3,1,,])[,20:1], col=gray(seq(0, 1, length.out=256)),
      axes=FALSE, asp=1)

## Data specific constants
num.class <- dim(training.data)[1]            # Number of classes (10)
num.training <- dim(training.data)[2]         # Number of all training data
d <- prod(dim(training.data)[3:4])            # Dimension (20x20=400)
num.test <- dim(test.data)[2]                # Number of test data
dim(training.data) <- c(num.class, num.training, d) # Reshape training data to 10*500*400 matrix
dim(test.data) <- c(num.class, num.test, d) # Same for test.
training.label <- rep(0:9, num.training) # Labels of training data.
test.label <- rep(0:9, num.test) # Labels of test data

## Part d
## --------------
##    Training
## --------------

## For this part, we pick a single digit class, and fit the mixture model for M=2,3,5,8

## create a vector denoting the class assignment
label_gamma <- function(m,M)
## m: the specified class
## M: total number of classes
{
  gamma <- rep(0, M)
  gamma[m] <- 1
  return(gamma)
}

## calculate the log-probability ln(p(X|\mu))
log_prob <- function(mu, X)
## mu: Bernoulli parameter
## X: observation data
{
  lp <- sum(X*log(mu) + (1-X)*log(1-mu))
  return(lp)
}

## get gamma for given Xi
gamma_xi <- function(X, pi, mu)
## X: data for a single observation
## pi: current pi hat
## mu: current mu hat
{
  M <- length(pi)
  l <- sapply(1:M, function(i) log_prob(mu[,i], X)+log(pi[i]))
```

```
  l_star <- max(l)
  gamma_xi <- sapply(1:M, function(i) exp(l[i]-l_star)/sum(exp(l-l_star)))
  return(gamma_xi)
}

## Calculate gamma for theta old
gamma_x <- function(X, pi, mu)
## X: full set of observations (N)
## pi: current pi hat
## mu: current mu hat
{
  N <- dim(X)[1]
  gamma <- sapply(1:N, function(i) gamma_xi(X[i,], pi, mu))
  gamma <- t(gamma)
}

## Update mu according to current gamma
mu_update <- function(gamma, X)
## gamma: current gamma(Z_im)
## X: full set of observations
{
  N = dim(gamma)[1]
  M = dim(gamma)[2]
  gamma_sum_m <- apply(gamma,2,sum)
  mu_mj <- function(m,j) (sum(gamma[,m]*X[,j])+1)/(gamma_sum_m[m]+2)
  all_dim <- function(m) sapply(1:d, function(j) mu_mj(m,j))
  mu_new <- sapply(1:M, all_dim)
  return(mu_new)
}

## Update pi accoridng to current gamma
pi_update <- function(gamma)
## gamma: current gamma(Z_im)
{
  N = dim(gamma)[1]
  M = dim(gamma)[2]
  sum_m <- (apply(gamma,2,sum) + 1)
  pi_new <- sum_m/sum(sum_m)
  return(pi_new)
}

## Calculate Q(theta,theta_old)+ln(p(theta)) as the threshold for stopping
threshold <- function(X, gamma, mu, pi)
## X: full set of observations
## Gamma: current gamma(Z_im)
## mu: current mu hat
## pi: current pi hat
{
  N <- dim(gamma)[1]
  M <- dim(gamma)[2]
  log_bern <- X%*%log(mu) + (1-X)%*%log(1 - mu)
  log_pi <- matrix(log(pi), nrow=N, ncol=M)
  Q <- sum(gamma * (log_pi + log_bern))
```

```
  log_p <- sum((log(pi)) + sum(log(mu) + log(1 - mu))) + M*d*log(6) + log(factorial(2*M-1))
  threshold <- Q + log_p
  return(threshold)
}

## Training with a single digit class
train_k <- function(k,M,X)
## k: the chosen digit class
## M: number of classes
## X: data
## return: list(mu,pi,M)
{
  e <- 0.001  ## threshold - stopping condition

  Z <- sample(1:M, num.training, replace=TRUE)  # create latent variable
  gamma <- sapply(1:num.training, function(i) label_gamma(Z[i], M))  ## initial gamma
  gamma <- t(gamma)

  # initial mu and pi
  mu_hat <- mu_update(gamma, X)
  pi_hat <- pi_update(gamma)

  threshold_old <- 0
  log_lik <- rep(0,100)  ## store the log-likelihood at each iteration

  for (i in 1:1000)
  {
    ## E-Step
    gamma <- gamma_x(X, pi_hat, mu_hat)
    ## M- STep
    mu_hat <- mu_update(gamma, X)
    pi_hat <- pi_update(gamma)
    ## compute threshold
    threshold <- threshold(X, gamma, mu_hat, pi_hat)
    log_lik[i] <- threshold


    if (abs(threshold - threshold_old) < e)
    {
      break
    }
    threshold_old <- threshold
  }

  result <- list(mu <- mu_hat, pi <- pi_hat, M <- M, log_lik <- log_lik[1:i])
}

## We pick the number 5 and see how this training process works

## reports the log-likelihood and estimated image
report <- function(k, M)
## k: chosen number digit
## M: number of classes
```

```
{
  parameters <- train_k(k,M,training.data[k+1,,])
  mu <- parameters[[1]]
  log_lik <- parameters[[4]]
  par(mfrow = c(1,M))
  for (i in 1:M)
  {
    im <- mu[,i]
    dim(im) <- c(20,20)
    image(t(1 - im)[,20:1], col=gray(seq(0, 1, length.out=256)),
          axes=FALSE, asp=1)
  }
  log_lik
}

report(5,2)
report(5,3)
report(5,5)
report(5,8)


## ------------------
##   Classification
## ------------------

## Compute the maximum likelihood with given parameters
max_lik <- function(X, pi, mu)
## X: test data
## pi: classifier pi
## mu: classifier mu
{
  M <- length(pi)
  single <- sapply(1:M, function(m) (pi[m] + log_prob(mu[,m], X)))
  return(sum(single))
}

## classification
classification <- function(ca,cb,X)
## ca: set of parameters a
## cb: set of parameters b
## X: test data
{
  max_lika <- max_lik(X, ca[[2]], ca[[1]])
  max_likb <- max_lik(X, cb[[2]], cb[[1]])
  return(max_lika >= max_likb)
}

## test the classifier with given digits a,b and M
test <- function(a, b, M)
## a: first digit
## b: second digit
## M: number of classes
{
```

```
  ca <- train_k(a,M,training.data[a+1,,])
  cb <- train_k(b,M,training.data[b+1,,])

  belong_a <- sapply(1:num.test, function(i) classification(ca, cb, test.data[a+1,i,]))
  belong_b <- sapply(1:num.test, function(i) classification(ca, cb, test.data[b+1,i,]))

  error_a <- sum(!belong_a)
  error_b <- sum(belong_b)

  error_rate <- (error_a+error_b)/(num.test*2)
  return(error_rate)
}


## Run classification tests

test1 <- test(0,1,5) ## digits 0,1 with M=5
test2 <- test(4,6,2) ## digits 4,6 with M=2
test3 <- test(3,8,3) ## digits 3,8 with M=3
```