STAT 24610 Final Project

Kevin O'Connor

June 3, 2016

1 LDA

Part a

First we will show that the Bayes' classifier has the form $h(x) = \text{argmax}_k \left\{ w_k^T x + w_{0k} \right\}$.

$$\begin{aligned}
h(x) &= \text{argmax}_k \left\{ P(C_k|x) \right\} \\
&= \text{argmax}_k \left\{ \frac{P(x|C_k)P(C_k)}{P(x)} \right\} \\
&= \text{argmax}_k \left\{ \frac{P(x|C_k)P(C_k)}{\sum\limits_k P(x,C_k)} \right\} \\
&= \text{argmax}_k \left\{ \frac{P(x|C_k)P(C_k)}{\sum\limits_k P(x|C_k)P(C_k)} \right\} \\
&= \text{argmax}_k \left\{ \frac{\pi_k \mathcal{N}(x{:}\mu_k,\Sigma)}{\sum\limits_k \pi_k \mathcal{N}(x{:}\mu_k,\Sigma)} \right\} \\
&= \text{argmax}_k \left\{ \frac{((2\pi)^p|\Sigma|)^{-1/2}\pi_k \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)}{\sum\limits_k ((2\pi)^p|\Sigma|)^{-1/2}\pi_k \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)} \right\} \\
&= \text{argmax}_k \left\{ \frac{\pi_k \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)}{\sum\limits_k \pi_k \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)} \right\} \\
&= \text{argmax}_k \left\{ \frac{\pi_k \exp\left(-\frac{1}{2}\left(x^T\Sigma^{-1}x - 2(\mu_k^T\Sigma^{-1})x + \mu_k\Sigma^{-1}\mu_k\right)\right)}{\sum\limits_k \pi_k \exp\left(-\frac{1}{2}\left(x^T\Sigma^{-1}x - 2(\mu_k^T\Sigma^{-1})x + \mu_k\Sigma^{-1}\mu_k\right)\right)} \right\} \\
&= \text{argmax}_k \left\{ \frac{\exp\left(-\frac{1}{2}x^T\Sigma^{-1}x\right)\pi_k \exp\left(\mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k\right)}{\exp\left(-\frac{1}{2}x^T\Sigma^{-1}x\right)\sum\limits_k \pi_k \exp\left(\mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k\right)} \right\} \\
&= \text{argmax}_k \left\{ \ln\left(\frac{\pi_k \exp\left(\mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k\right)}{\sum\limits_k \pi_k \exp\left(\mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k\right)} \right) \right\} \\
&= \text{argmax}_k \left\{ \mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k + \ln(\pi_k) - \ln\left(\sum_k \pi_k \exp\left(\mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k\right) \right) \right\}
\end{aligned}$$

Now, we can drop the $\ln\left(\sum\limits_k \pi_k \exp\left(\mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k\right) \right)$ term since it will be the same for all k, given some x. And we can write our final classifier as

$$h(x) = \text{argmax}_k \left\{ \mu_k^T\Sigma^{-1}x - \frac{1}{2}\mu_k\Sigma^{-1}\mu_k + \ln(\pi_k) \right\}$$

Thus, we have

$$w_k = \Sigma^{-1}\mu_k$$

$$w_{0k} = -\tfrac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \ln(\pi_k)$$

Part b

We will derive the maximum likelihood estimators for μ_k and Σ. First we have our likelihood function,

$$\begin{aligned}\mathcal{L}(\theta;\mathcal{X}) &= \textstyle\prod_{i=1}^n P(X_i|\theta) \\ &= \textstyle\prod_{i=1}^n \frac{1}{\sqrt{(2\pi)^p|\Sigma|}} \exp\left(-\frac{1}{2}(X_i-\mu_{y_i})^T\Sigma^{-1}(X_i-\mu_{y_i})\right)\end{aligned}$$

And thus a log-likelihood,

$$l(\theta;\mathcal{X}) = \sum_{i=1}^n \left(-\frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(X_i-\mu_{y_i})^T\Sigma^{-1}(X_i-\mu_{y_i})\right)$$

Now, maximizing with respect to μ_k,

$$\begin{aligned}\tfrac{d}{d\mu_k} l(\theta;\mathcal{X}) &= \tfrac{d}{d\mu_k}\sum_{i=1}^n \left(-\tfrac{p}{2}\ln(2\pi) - \tfrac{1}{2}\ln|\Sigma| - \tfrac{1}{2}(X_i-\mu_{y_i})^T\Sigma^{-1}(X_i-\mu_{y_i})\right) \\ &= \tfrac{d}{d\mu_k}\sum_{i:y_i=k}\left(-\mu_k^T\Sigma^{-1}X_i + \tfrac{1}{2}\mu_k^T\Sigma^{-1}\mu_k\right) \\ &= \sum_{i:y_i=k}\left(\Sigma^{-1}X_i + \Sigma^{-1}\mu_k\right) \\ &= \Sigma^{-1}\left(n_k\mu_k - \sum_{i:y_i=k} X_i\right) \\ &\equiv 0\end{aligned}$$

Where we have defined $n_k = \sum_{i=1}^n \mathbb{1}\{y_i = k\}$. From here, we can see that the maximum likelihood estimator for μ_k is given by

$$\hat{\mu}_k = \frac{1}{n_k}\sum_{i:y_i=k} X_i$$

Now maximizing with respect to Σ^{-1}, we first rewrite the log-likelihood using the trace.

$$\begin{aligned}l(\theta;\mathcal{X}) &= \sum_{i=1}^n \left(-\tfrac{p}{2}\ln(2\pi) - \tfrac{1}{2}\ln|\Sigma| - \tfrac{1}{2}(X_i-\mu_{y_i})^T\Sigma^{-1}(X_i-\mu_{y_i})\right) \\ &\propto -\tfrac{n}{2}\ln|\Sigma| - \tfrac{1}{2}\sum_{i=1}^n \mathrm{Tr}\left(\Sigma^{-1}(X_i-\mu_{y_i})(X_i-\mu_{y_i})^T\right) \\ &= -\tfrac{n}{2}\ln|\Sigma| - \tfrac{1}{2}\mathrm{Tr}\left(\Sigma^{-1}\sum_{i=1}^n (X_i-\mu_{y_i})(X_i-\mu_{y_i})^T\right)\end{aligned}$$

Now, differentiating with respect to Σ^{-1},

$$\begin{aligned}\tfrac{d}{d\Sigma^{-1}}\tilde{l}(\theta;\mathcal{X}) &= \tfrac{d}{d\Sigma^{-1}}\left(-\tfrac{n}{2}\ln|\Sigma| - \tfrac{1}{2}\mathrm{Tr}\left(\Sigma^{-1}\sum_{i=1}^n (X_i-\mu_{y_i})(X_i-\mu_{y_i})^T\right)\right) \\ &= \tfrac{n}{2}\Sigma - \tfrac{1}{2}\sum_{i=1}^n (X_i-\mu_{y_i})^T(X_i-\mu_{y_i}) \\ &\equiv 0\end{aligned}$$

Thus, we find the maximum likelihood estimator for Σ is given by

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (X_i-\mu_{y_i})(X_i-\mu_{y_i})^T$$

Part c

Next we are asked to consider a regularized form of the sample covariance matrix, given by

$$\hat{\Sigma}_\lambda = (1-\lambda)\hat{\Sigma} + (\lambda/4)I_p$$

for $\lambda \in (0,1)$. We can see why $\hat{\Sigma}_\lambda$ must be invertible by considering how the transformation affects the eigenvalues. First, by multiplying $\hat{\Sigma}$ by $(1-\lambda)$, any eigenvalue of $\hat{\Sigma}$, λ_i, becomes $(1-\lambda)\lambda_i$. Then, by adding $(\lambda/4)I_p$ to the diagonals, we are transforming the characteristic equation from $\det((1-\lambda)\hat{\Sigma} - k) = 0$ to $\det\Big((1-\lambda)\hat{\Sigma} - (k-\lambda/4)\Big) = 0$. So, all eigenvalues, $(1-\lambda)\lambda_i$ are shifted up by $\lambda/4$. Thus, if $\hat{\Sigma}$ has an eigenvalue λ_i, then $\hat{\Sigma}_\lambda$ has the eigenvalue $(1-\lambda)\lambda_i + \lambda/4$. It is clear that

$$(1-\lambda)\lambda_i + \lambda/4 > \lambda_i$$

Since $\hat{\Sigma}$ is positive semi-definite and therefore any $\lambda_i \geq 0$. So, this transformation increases the eigenvalues, yielding a larger determinant since the determinant is the product of the eigenvalues. This then helps us deal with situations when $\hat{\Sigma}$ is sparse and thus has a determinant that is close to zero, meaning that it is nearly singular. Furthermore, we add only $\lambda/4$ so as to preserve the information contained in $\hat{\Sigma}$ to some extent. Otherwise, for values of λ close to one, we would find that the covariance matrix was approximately equal to the identity and was thus uninformative.

Part d

Next we are asked to implement LDA as we have outlined in Parts a-c. Below, I walk you through my implementation, written in *R*.

We first load in the data using the code provided.

```
## Loading data, setting up for use in R classification packages.
setwd("/Users/kevinoconnor/Documents/School/STAT246/")
library(MASS)
load('digits.RData')
num.class <- dim(training.data)[1] # Number of classes
num.training <- dim(training.data)[2] # Number of training data per class
d <- prod(dim(training.data)[3:4]) # Dimension of each training image (r*c)
num.test <- dim(test.data)[2] # Number of test data
dim(training.data) <- c(num.class*num.training, d) # Reshape training data to 2-dim 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
```

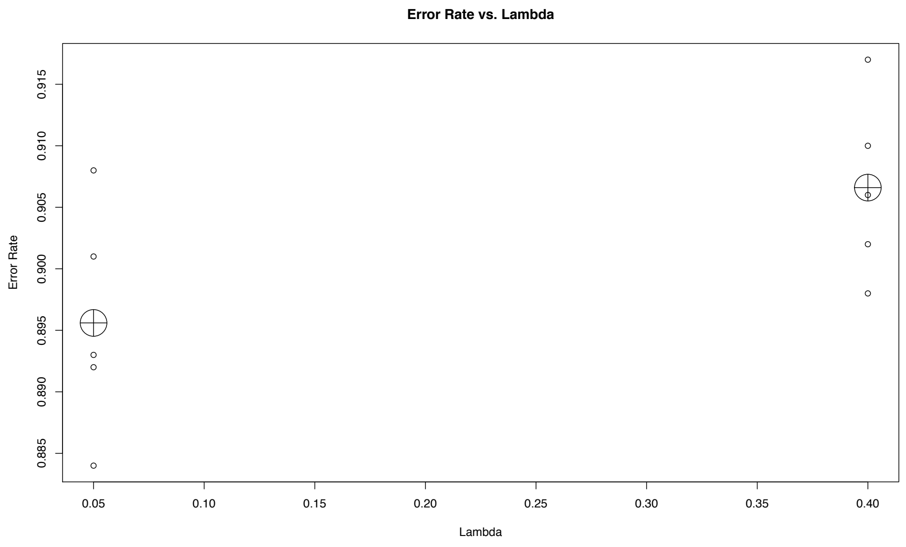
Next, we write the methods that perform the classification, using the linear decision boundary and our formula for w_k and w_{0k}.

```
## Compute Sigma
compute_sigma <- function(mu, labels, x){
     total = (x[1,]-mu[[labels[1]+1]])%*%t(x[1,]-mu[[labels[1]+1]]) # initialize
     for (i in 2:length(x[,1])){ # loop through samples
     total = total + ((x[i,]-mu[[labels[i]+1]])%*%t(x[i,]-mu[[labels[i]+1]])) # add to sum
     }
     return((1/length(x[,1]))*total) # divided by n
}

## Covariance matrix regularization and inversion method.
inv_reg_covmat <- function(sigma, lambda){
     sigma_l = (1-lambda)*sigma + (lambda/4)*diag(length(sigma[1,])) # regularize as
         suggested
     return(ginv(sigma_l)) # return inverse of regularized matrix
}

## Bayes Classification Method
### Returns the predicted digit based on Bayes classification as specified in Part a.
### Requires mu as a list of vectors, sigma_inv as a matrix, p as a vector, and x (data)
     as a vector.
bayes_classify <- function(mu, sigma_inv, p, x){
     w_0 = c(); w=list(); vals = c() # initializing
     for (i in 1:10){
          w_0 = c(w_0, -(1/2)*(mu[[i]] %*% sigma_inv %*% mu[[i]]) + log(p[i])) # intercept
               for i'th class
          w[[i+1]] = sigma_inv %*% mu[[i]] # slope for i'th class
          vals = c(vals, t(w[[i+1]]) %*% x + w_0[i]) # storing w_0i + w_i*x
     }
     return(which(vals==max(vals))-1) # returning class which maximizes w_0i + w_i*x
}

## Test classifier
### Returns error rate and predicted classes for a set of test data.
### Requires mu as a list of vectors, sigma as a matrix, p as a vector, lambda as a
     number, x as a matrix, and labels as a vector.
test_classifier <- function(mu, sigma, p, lambda, x, labels){
     sig_inv = inv_reg_covmat(sigma, lambda) # regularize and invert covariance matrix
     classes = c() # initialize
     for (i in 1:length(x[,1])){ # loop through examples in test set x
          classes = c(classes, bayes_classify(mu, sig_inv, p, x[i,])) # append to vector
               of predicted classes
     }
     error_rate = 1-mean((labels == classes)+0) # compute error rate
     return(list(error_rate, classes)) # return error rate and predicted classes
}
```

With these methods written, we may perform cross-validation. As suggested, a random subset of 400 training examples from each class were used to train the classifier at each value of λ and then classification was tested on the remaining 100 examples from each class. This was performed 5 times at each λ, where we let $\lambda = \{0.05, 0.10, 0.15, ...0.95\}$. The code written for this can be found below.

```
## Cross validation
lambda = seq(0.05, 0.95, 0.05)
p = rep(0.1, 10)
errors = list()
for(i in 1:length(lambda)){
    error_i = c()
    for (j in 1:5){
        mu = list()
        cv.train.data = list()
        cv.test.data = list()
        cv.test.labels = list()
        for (k in 0:9){
            inds = sample(500,400)
            cv.train.data[[k+1]] = training.data[which(training.label==k),][inds,]+0
            cv.test.data[[k+1]] = training.data[which(training.label==k),][-inds,]+0
            cv.test.labels[[k+1]] = training.label[which(training.label==k)][-inds]
            mu[[k+1]] = colMeans(cv.train.data[[k+1]])
        }
        sigma = cov(matrix(unlist(cv.train.data), nrow=4000, byrow=T))
        cv.test.data = matrix(unlist(cv.test.data), nrow=1000, byrow=T)
        cv.test.labels = unlist(cv.test.labels)
        error_i = c(error_i, test_classifier(mu, sigma, p, lambda[i], cv.test.data,
            cv.test.labels)[[1]])
    }
    errors[[i]] = error_i
}
```

The error rate was recorded at each step and can be seen in the plot below, with the average for each λ marked with a cross-hair. Numerical values are given below that.



```
> error_mat = matrix(unlist(errors), nrow=5, byrow=F)
> colnames(error_mat) = lambda
> error_mat
      0.05   0.1  0.15   0.2  0.25   0.3  0.35   0.4  0.45   0.5
[1,] 0.072 0.106 0.089 0.102 0.108 0.078 0.085 0.084 0.098 0.096
[2,] 0.103 0.112 0.099 0.114 0.080 0.086 0.066 0.076 0.101 0.098
[3,] 0.091 0.091 0.081 0.105 0.093 0.121 0.094 0.081 0.110 0.099
[4,] 0.113 0.103 0.108 0.117 0.085 0.087 0.073 0.104 0.096 0.082
[5,] 0.108 0.097 0.101 0.102 0.099 0.084 0.113 0.099 0.099 0.090
      0.55   0.6  0.65   0.7  0.75   0.8  0.85   0.9  0.95
[1,] 0.099 0.089 0.097 0.092 0.096 0.092 0.096 0.110 0.081
[2,] 0.103 0.091 0.098 0.104 0.100 0.099 0.095 0.099 0.101
[3,] 0.097 0.087 0.096 0.100 0.104 0.105 0.094 0.094 0.092
[4,] 0.103 0.118 0.109 0.096 0.093 0.098 0.090 0.090 0.095
[5,] 0.079 0.090 0.105 0.103 0.105 0.096 0.103 0.091 0.096
> colMeans(error_mat)
  0.05    0.1   0.15    0.2   0.25    0.3   0.35    0.4   0.45
0.0974 0.1018 0.0956 0.1080 0.0930 0.0912 0.0862 0.0888 0.1008
   0.5   0.55    0.6   0.65    0.7   0.75    0.8   0.85    0.9
0.0930 0.0962 0.0950 0.1010 0.0990 0.0996 0.0980 0.0956 0.0968
  0.95
0.0930
```

Though $\lambda = 0.35$ yields the minimum average error rate, its value seems to be fairly unstable. We will fix $\lambda = 0.40$ which gives the second lowest average error rate but also exhibits a bit more stability.

Now, we have our classifier completely specified. Specifically,

$$h(x) = \text{argmax}_k \left\{ \mu_k^T \Sigma_\lambda^{-1} x - \frac{1}{2} \mu_k \Sigma_\lambda^{-1} \mu_k + \ln(\pi_k) \right\}$$

where $\Sigma_\lambda = 0.6\Sigma + 0.1 I_p$. We may now proceed to test the classifier on the test set. First, it will be trained on the entire training set and then the same procedure will be followed as before in testing the classifier on the test data.

2 Mixture of Independent Bernoullis

Next we consider a mixture of independent Bernoulli distributions.

Part a

We place a Beta$(2,2)$ prior on the $\mu_{m,j}$ parameters and a Dirichlet$(2, ..., 2)$ prior on the π_m parameters and are asked to derive the EM algorithm for the MAP of $\mu_{m,j}$ and π_m.

1: Initialization

We begin by initializing $\mu_{m,j}$ and π_m. Assume this has been done for now, as a method of doing so will be addressed in Part b.

2: E Step

Next we evaluate $Q(\theta, \theta^{\text{old}})$. We begin by writing $p(X, Z|\theta)$.

$$p(X, Z|\mu, \pi) = \prod_{i=1}^{n} \prod_{m=1}^{M} \left(\pi_m \prod_{j=1}^{D} \mu_{m,j}^{X_{ij}} (1 - \mu_{m,j})^{1-X_{ij}} \right)^{z_{im}}$$

And thus the log is given by

$$\ln\left(p(X, Z|\mu, \pi)\right) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left(\ln(\pi_m) + \sum_{j=1}^{D} \Big[X_{ij} \ln(\mu_{m,j}) + (1 - X_{ij}) \ln(1 - \mu_{m,j}) \Big] \right)$$

Then the expectation of this over Z is given by

$$\mathbb{E}_Z\left[\ln\left(p(X, Z|\mu, \pi)\right)\right] = \sum_{i=1}^{n} \sum_{m=1}^{M} \gamma(z_{im}) \left(\ln(\pi_m) + \sum_{j=1}^{D} \Big[X_{ij} \ln(\mu_{m,j}) + (1 - X_{ij}) \ln(1 - \mu_{m,j}) \Big] \right)$$

where

$$\begin{aligned} \gamma(z_{im}) &= \mathbb{E}\left[z_{im}\right] \\ &= \frac{\sum_{z_i} z_{im} \prod_{k'} (\pi_{k'} p(X_i|\mu_{k'}))^{z_{ik'}}}{\sum_{z_i} \prod_{m} (\pi_m p(X_i|\mu_m))^{z_{im}}} \\ &= \frac{\pi_m p(X_n|\mu_m)}{\sum_m \pi_m p(X_i|\mu_m)} \end{aligned}$$

3: M Step

Next we seek to maximize the MAP,

$$\mathbb{E}_Z\left[\ln\left(p(X, Z|\mu, \pi)\right)\right] + \ln\left(p(\theta)\right)$$

We will first write $p(\theta) = p(\pi)p(\mu)$.

$$p(\theta) = \frac{1}{6\Gamma(2D)} \prod_{m=1}^{M} \prod_{j=1}^{D} \mu_{m,j}(1 - \mu_{m,j}) \prod_{n=1}^{M} \pi_n$$

And thus

$$\ln\left(p(\theta)\right) = \sum_{m=1}^{M} \sum_{j=1}^{D} \Big[\ln(\mu_{m,j}) + \ln(1 - \mu_{m,j}) \Big] + \sum_{m=1}^{M} \ln(\pi_m) + C$$

where C is a constant. Label the MAP, Q_{MAP}. Based on the work in the E-step, we have

$$\begin{aligned} Q_{\text{MAP}} = &\sum_{i=1}^{n} \sum_{m=1}^{M} \gamma(z_{im}) \left(\ln(\pi_m) + \sum_{j=1}^{D} \Big[X_{ij} \ln(\mu_{m,j}) + (1 - X_{ij}) \ln(1 - \mu_{m,j}) \Big] \right) \\ &+ \sum_{m=1}^{M} \sum_{j=1}^{D} \Big[\ln(\mu_{m,j}) + \ln(1 - \mu_{m,j}) \Big] + \sum_{m=1}^{M} \ln(\pi_m) + C \end{aligned}$$

Now, we maximize with respect to θ^{old}. Differentiating with respect to $\mu_{m,j}$,

$$\frac{d}{d\mu_{m,j}} Q_{\text{MAP}} = \sum_{i=1}^{n} \gamma(z_{im}) \left[X_{ij}\mu_{m,j}^{-1} - (1 - X_{ij})(1-\mu_{m,j})^{-1} \right] + \mu_{m,j}^{-1} - (1-\mu_{m,j})^{-1} \equiv 0$$

This gives us the solution,

$$\mu_{m,j}^{\text{new}} = \frac{1}{2+n_m} \left(1 + \sum_{i=1}^{n} \gamma(z_{im}) X_{ij} \right)$$

where

$$n_m = \sum_{i=1}^{n} \gamma(z_{im})$$

Now, before differentiating with respect to π_m, we can write Q_{MAP} with a Lagrange multiplier to account for the condition that $\sum\limits_{m=1}^{M} \pi_m = 1$. Now we will be differentiating,

$$Q_{\text{MAP}} + \lambda \left(\sum_m \pi_m - 1 \right)$$

Note this can be done since the term in parentheses is constrained to be 0. Now, differentiating with respect to π_m.

$$\frac{d}{d\pi_m} \left[Q_{\text{MAP}} + \lambda \left(\sum_m \pi_m - 1 \right) \right] = \sum_{i=1}^{n} \gamma(z_{im}) \pi_m^{-1} + \pi_m^{-1} + \lambda \equiv 0$$

This yields the solution,

$$\pi_m^{\text{new}} = \frac{-\left(\sum\limits_{i=1}^{n} \gamma(z_{im}) + 1 \right)}{\lambda} = \frac{-(n_m + 1)}{\lambda}$$

We can solve for λ by summing π_m over m and setting this equal to 1.

$$\begin{aligned} \sum_{m=1}^{M} \pi_m &= \sum_{m=1}^{M} \frac{-\left(\sum_{i=1}^{n} \gamma(z_{im}) + 1 \right)}{\lambda} \\ &= -\tfrac{1}{\lambda} \left[\sum_{m=1}^{M} \sum_{i=1}^{n} \gamma(z_{im}) + M \right] \\ &= \tfrac{1}{\lambda} [n + M] \\ &\equiv 1 \end{aligned}$$

This yields $\lambda = -(M+n)$. Thus, we find that

$$\pi_m^{\text{new}} = \frac{n_m + 1}{M + n}$$

4: Check for Convergence

As stated, we check that whether the convergence criteria have been satisfied. If not, repeat the iteration.

Part b

It is suggested that μ and π be initialized by assigning each example at random to one of M components. With this complete data, we may estimate $\mu_{m,j}$ as

$$\mu_{m,j} = \frac{1}{n_m} \sum_{i:X_i \in m} X_{ij}$$

where n_m is defined as before. Similarly, we may estimate π_m as

$$\pi_m = \frac{n_m}{n}$$

Then the EM algorithm may proceed as outlined in Part a.

Part c

Whenever we take the product of many numbers on a computer, there is a risk of exceeding the dynamic range of the computer. I.e., the product may become so small or large that the computer is not able to represent it in any of its standard numerical data types. By taking the log and exponentiating later, we can prevent such overflow in most cases.