**4.1**

*Which lines correspond to the E-step, which to the M-step?*

The following lines correspond to the E-step:

phiA = **dnorm**(yvar, mean = muA, sd = sdA)

phiB = **dnorm**(yvar, mean = muB, sd = sdB)

pA = lambda **\*** phiA

pB = (1 **-** lambda) **\*** phiB

ptot = pA **+** pB

pA = pA **/** ptot

pB = pB **/** ptot

The following lines correspond to the M-step:

lambda = **mean**(pA)

muA = **weighted.mean**(yvar, pA)

muB = **weighted.mean**(yvar, pB)

sdA = **sqrt**(**weighted.mean**((yvar **-** muA)**^**2, pA))

sdB = **sqrt**(**weighted.mean**((yvar **-** muB)**^**2, pB))

*What does the M-step do, what does the E-step do?*

In the soft-label EM algorithm, the E-step uses Bayes’ Rule to gauge the probability that data point belongs to group using the metric . Specifically:

[1]

The M-step then uses the updated s to update parameters of the gaussian distribution following the maximum-likelihood estimation of mean and variance. Specifically:

; ; [1]

The EM algorithm can be proven to converge to at least a local maximum using Jensen’s inequality [1].

*Why do we need to compute loglik?*

Because we need to compare the new and old log likelihood and calculate their difference. If the difference is smaller than the tolerance level, then we halt the program and conclude the algorithm has converged. This protects us from running an infinite loop.

*Compare the result of what we are doing here to the output of the normalmixEM function from the mixtools package.*

The result from *normalmixEM matches that from the textbook:*

A picture containing knife, table

Description automatically generated

R Code:

*#4.1*

*library(mixtools)*

*yvar <- readRDS("../data/Myst.rds")$yvar*

*results <- mixtools::normalmixEM(yvar)*

*t(data.frame(results$mu,results$sigma))*

**4.2**

A screenshot of a cell phone

Description automatically generated

As shown above, the two distributions match closely.

R Code:

*#original distribution*

*lambda <- rgamma(10000, shape = 10, rate = 3/2)*

*distribution1 <- rpois(length(lambda), lambda = lambda)*

*#theoretical distribution generated from parameters obtained from fitting in 4.4.3*

*p\_size <- 10.06276*

*p\_prob <- 0.6038492*

*distribution2 <- rnbinom(10000, p\_size, p\_prob)*

*qqplot(distribution1,distribution2, asp = 1, xlab = "simulated", ylab = "theoretical")*

*abline(a = 0, b = 1, col="blue")*

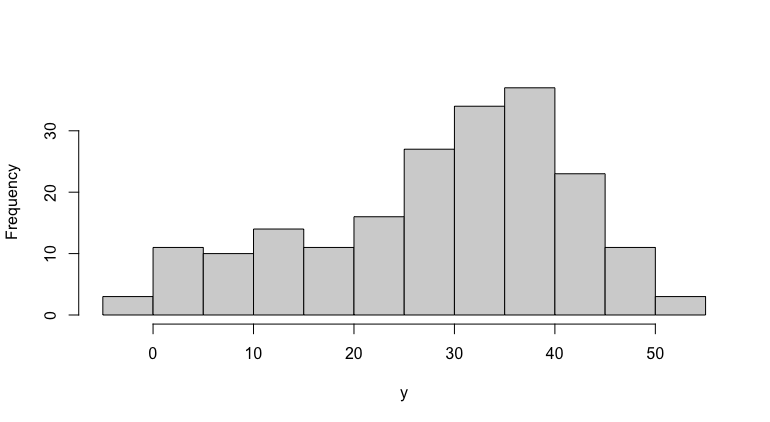
**4.3**

*(a)*

A screenshot of a cell phone

Description automatically generated

From plotting data points from the two classes separately, it is evident that data points from *class = 1* have a linear relationship and data points from *class = 1* have a semi-quadratic relationship.



From the histogram of *yn* alone, it is difficult to reach the conclusion that the population is made of two mixtures.

*(b)*

m1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)

m1

*(c)*

The estimated parameters of the mixture components are as follows:

A picture containing bird, flower

Description automatically generated

This matches our expectation from observing the plot in part *(a)*. The blue part is component 1 and will have its intercept at -0.2099. Its linear coefficient is significantly larger than its quadratic counterpart, hence it is mostly linear in shape. On the other hand, the green part is component 2 and will have its intercept at 14.7165. Its quadratic coefficient is non-negligible and its negative value further confirms its concave shape.

The truth table is as follows:

A close up of text on a white background

Description automatically generated

The clustering algorithm correctly matched 95% of the datapoints from *mean(abs(truthTable$class - truthTable$cluster))*

The summary of object *m1* shows that the log likelihood is negative enough to reject the null hypothesis that the two clusters come from the same underlying distribution. In other words, we can be fairly certain that the data is made of two distinct mixtures generated from .

A picture containing bird, flower

Description automatically generated

*(d)*

A picture containing photo, white

Description automatically generated

As shown in the plot above, the clustering matches closely to what is reported in part *(a)*.

R Code:

*#4.3*

*library("flexmix")*

*data("NPreg")*

*#a*

*NPreg1 <- dplyr::filter(NPreg,class==1)*

*NPreg2 <- dplyr::filter(NPreg,class==2)*

*plot(NPreg1$x,NPreg1$yn,col="blue",xlab = "x", ylab = "y")*

*points(NPreg2$x,NPreg2$yn,col="green")*

*hist(NPreg$yn,main = "",xlab="y")*

*#b*

*m1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)*

*m1*

*#c*

*m1@components*

*NPreg\_clustered <- data.frame(NPreg, "cluster" = as.factor(clusters(m1)))*

*truthTable <- NPreg\_clustered[,c("x","yn","class","cluster")]*

*class(truthTable)*

*mean(abs(truthTable$class - truthTable$cluster))*

*summary(m1)*

*#d*

*library("ggplot2")*

*ggplot(data = truthTable, aes(x = x, y = yn)) + geom\_point(aes(color=cluster)) + labs(x="x",y="y")*

References

[1] A. Ng, “Lecture 14 - Expectation-Maximization Algorithms | Stanford CS229: Machine Learning (Autumn 2018) - YouTube,” 2017. [Online]. Available: https://www.youtube.com/watch?v=rVfZHWTwXSA. [Accessed: 14-Jun-2020].