Statistical Computing Project 4

Mario Ibanez

March 26, 2016

Introduction

The purpose of this report is to evaluate the effectiveness of the EM algorithm in solving a mixture model problem. The problem that will be solved in this report is that given a data set, attempt to estimate the proportion at which data was sampled from two normal distributions along with estimating the means and standard deviations of those two distributions. More explicitly, the data will be sampled from the distribution:

$$f(x) = w_1 N(\mu_1, \sigma_1) + w_2 N(\mu_2, \sigma_2) = w_1 f_1(x) + w_2 f_2(x)$$

and the EM algorithm will be used to estimate the parameters w_1, w_2, μ_1, μ_2, σ_1 and σ_2. In addition to estimating these parameters, an attempt can be made at evaluating the algorithm itself by calculating various statistics reflecting the algorithm's performance.

Methods

In order to evaluate the algorithm, known values for the parameters listed above will be used to generate data. The algorithm will then use this data to estimate the six unknown mixture model parameters. However, because of the constraint that $w_1 + w_2 = 1$, there are in fact only five unknown parameters. If the true values for these parameters are known, the algorithm can be evaluated for effectiveness by comparing the estimates it generates to the true values. A method that will be used to evaluate the accuracy of the algorithm will be "z-scores" for the estimated means. Explicitly:

$$z_1 = \frac{\hat{\mu_1} - \mu_1}{\sigma_1} \text{ and } z_2 = \frac{\hat{\mu_2} - \mu_2}{\sigma_2}$$

These z-scores will measure how far away from the true values the estimates of the means were.

Though there are different approaches possible in order to evaluate the EM algorithm's ability to solve a mixture model problem, the one used in this report is to run the algorithm repeatedly for different samples from the same distribution. This will give an idea, for that specific distribution, how the algorithm tends to perform. Another alternative approach that will not be taken in this report is to run the algorithm a single time for different distributions and compare the algorithm's ability to solve the problem depending on the parameters.

Data

The data used in this report to evaluate the EM algorithm will be 100 values sampled from the following mixture distribution:

$$f(x) = (0.30)N(-1, 2) + (0.70)N(2, 1)$$

In order to perform this sampling, a random number will be sampled from the binomial distribution with parameters $n = 100$ and $p = 0.30$. This value will be used to determine how many values are sampled from $N(-1, 2)$. The rest of the values will then be sampled from $N(2, 1)$. Sampling from a normal distribution with any parameters is easy in the *R* language using a built in function. The code below generates the data:

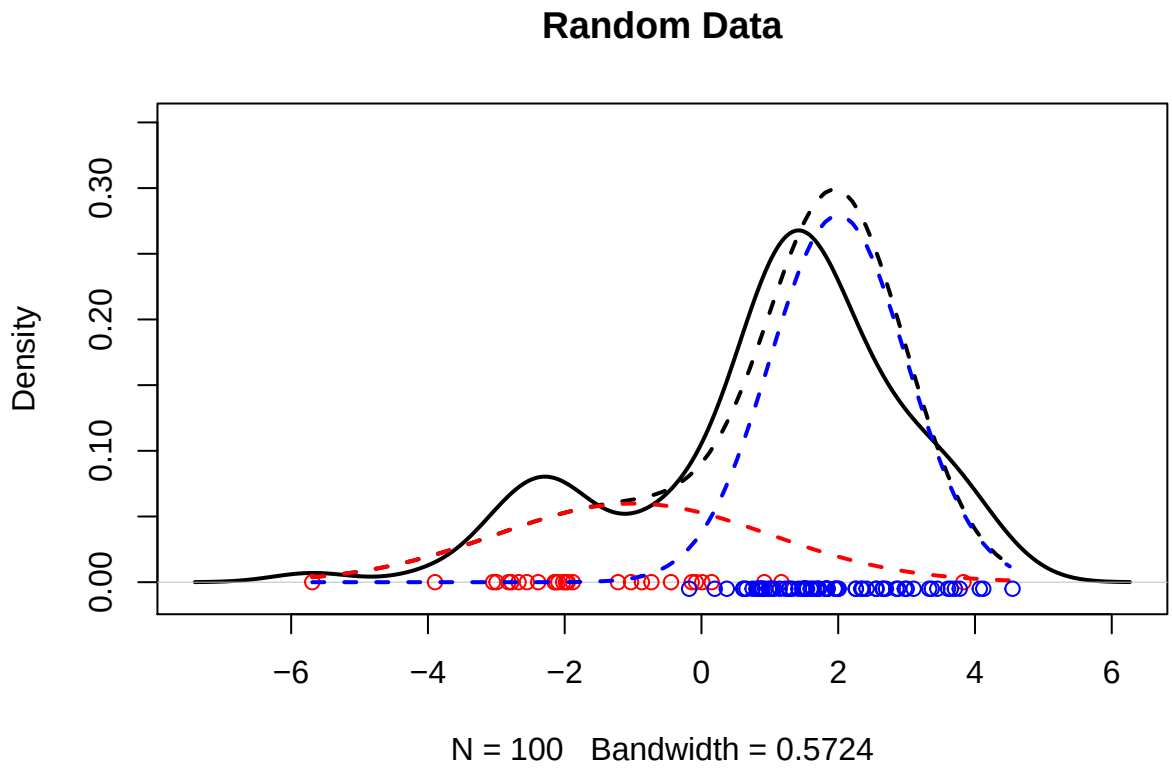
```r
# Mixture weights, seed, and total sample size
proportion <- 0.30
sample_size <- 100
set.seed(1234)

# Means and standard deviations
mu_1 <- -1; mu_2 <- 2
sigma_1 <- 2; sigma_2 <- 1

# Calculate how many values will come from each component
sample1_size <- rbinom(n = 1, size = sample_size, prob = proportion)
sample2_size <- sample_size - sample1_size

# Sample that many values from each respective normal distribution
sample1 <- rnorm(n = sample1_size, mean = mu_1, sd = sigma_1)
sample2 <- rnorm(n = sample2_size, mean = mu_2, sd = sigma_2)
```

The plot below allows us to visualize the data set that will be used later in the report to evaluate the EM algorithm:



In the plot, the dotted lines represent the true population distributions. The red dotted line is the PDF of $N(-1, 2)$ while the blue dotted line is the PDF of $N(2, 1)$ and the black dotted line is the PDF of $(0.30)N(-1, 2) + (0.70)N(2, 1)$. The red and blue points at the bottom of the plot represent the generated

data. Finally, the black solid line is a plot of the density of the generated data. Notice of course that while the black dotted line and the black solid line should be similar in shape, there is no guarantee that from sample to sample this will occur. This is especially true when the two distributions significantly overlap as they do in the example in this report.

EM Algorithm

The EM algorithm works by alternating between two steps until convergence. The first step is the Expectation step and the second step is the Maximization step. In this applicaton of the algorithm, the expectation step involves the relative probability that a data point was generated from the first distribution. (Note, the first distribution is $f_1(x) = N(-1, 2)$) This value is calcuated for each data point in each iteration. The formula for the i^{th} value is:

$$\hat{z_i} = \frac{\hat{w_1} f_1(x_i)}{\hat{w_1} f_1(x_i) + (1 - \hat{w_1}) f_2(x_i)}$$

This value is used to weight the estimates for the parameters μ_1, μ_2, σ_1, and σ_2. The next step is the maximization step. Updated estimates of the five parameters are calculated:

$$\hat{\mu_1} = \frac{\sum \hat{z_i} x_i}{\sum \hat{z_i}} \text{ and } \hat{\mu_2} = \frac{\sum (1 - \hat{z_i}) x_i}{\sum (1 - \hat{z_i})}$$

$$\hat{\sigma_1^2} = \frac{\sum \hat{z_i} (x_i - \hat{\mu_1})^2}{\sum \hat{z_i}} \text{ and } \hat{\sigma_2^2} = \frac{\sum (1 - \hat{z_i}) (x_i - \hat{\mu_2})^2}{\sum (1 - \hat{z_i})}$$

$$\hat{w_1} = \frac{\sum \hat{z_i}}{N} \text{ where } N = 100$$

Written more explictily, the steps are:
1) Initialize the five unknown paramater values with guesses
2) Calculate $\hat{z_i}$ for each value
3) Calculate new esimates for the five parameters
4) Return to step 2 if convergence criteria has not been met

The algorithm will run until the difference between to successive estimates of μ_1 differ by less than 0.01. This is an arbitrary convergence criteria but will serve fine for the purpose of this report.

Simulation

As was partly mentioned earlier in the paper, the EM algorithm will be evaluated by repeatedly generating new random data from the distribution:

$$f(x) = (0.30)N(-1, 2) + (0.70)N(2, 1)$$

The "z-scores" defined earlier will be calculated each time. Random data will be generated 500 times and the algorithm allowed to run until reaching its convergence criteria each time. The initial values for the parameters will be the same in each trial: $\mu_1 = -0.5$, $\mu_2 = 0.5$, $\sigma_1 = 1$, $\sigma_2 = 1$ and $w_1 = 0.5$. The simulations will be done in R by using a loop to repeatedly generate random data, rerun the algorithm, and store the z scores for that trial. The code in the appendix accomplishes this.

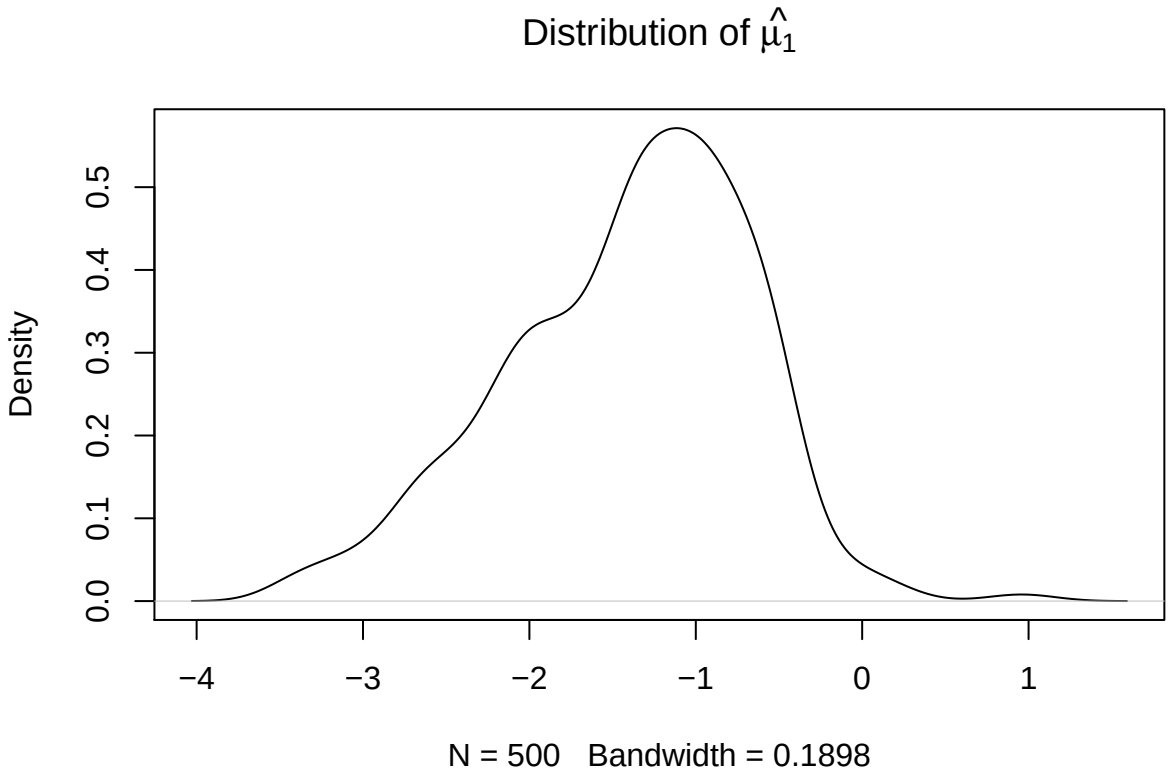
Next, the results of the simulation are analyzed.

Results

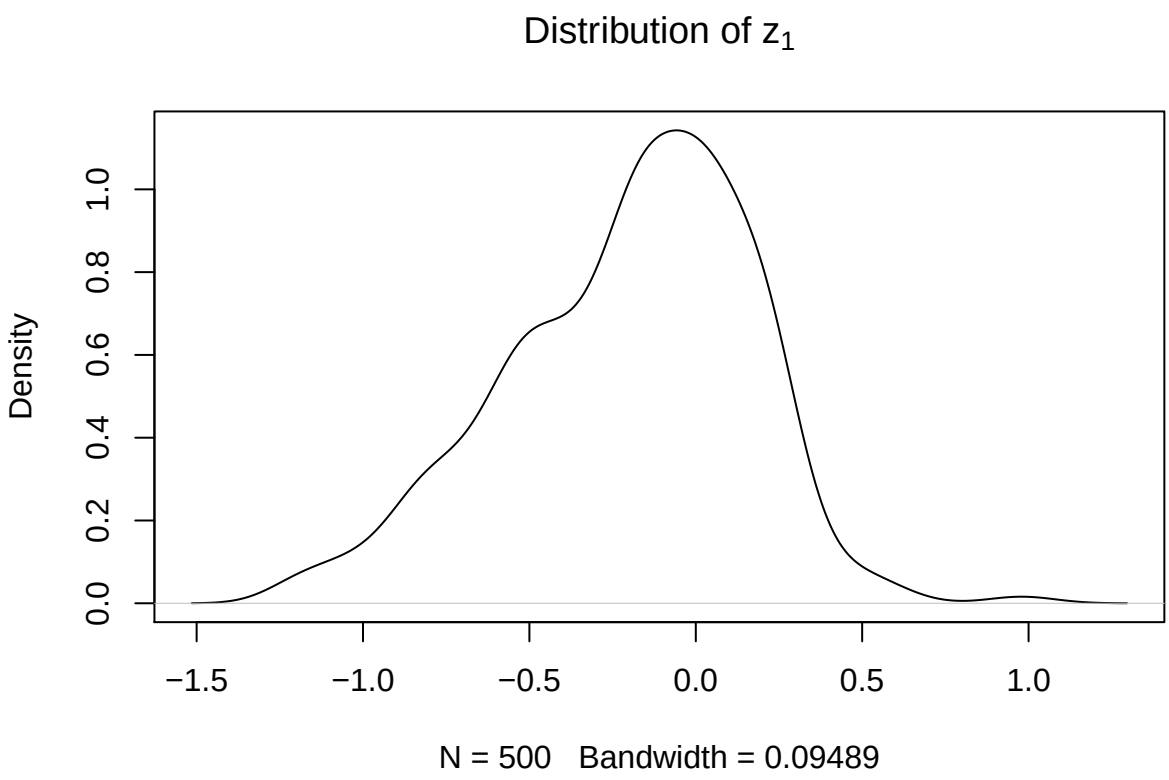
Again, the simulation repeated the EM algorithm 500 times and each time various values were stored. We can look at these results value by value to understand what occured.

Estimate of μ_1

The true value of μ_1 is -1. The concern with this parameter is that since this distribution is being sampled from with a weight of 0.30 and has a high standard deviation of 2, it can be difficult to detect its true value. The plot below shows a distribution of the estimates $\hat{mu}_1$:



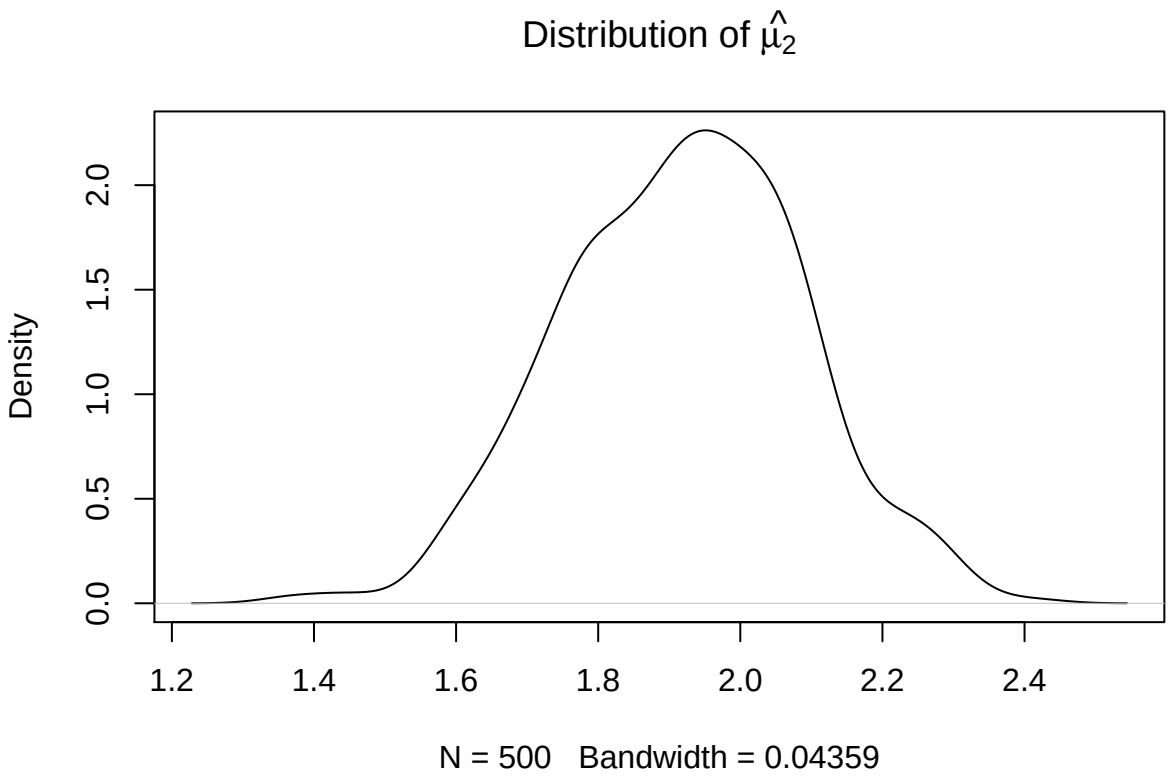
We see that the algorithm does a fairly good job estimating μ_1. The mode of the distribution is around -1, and the left tail is heavy. The reason for this is that the other component of the mixture is to the right of this component. The plot below is of the "z-scores" of the estimates:



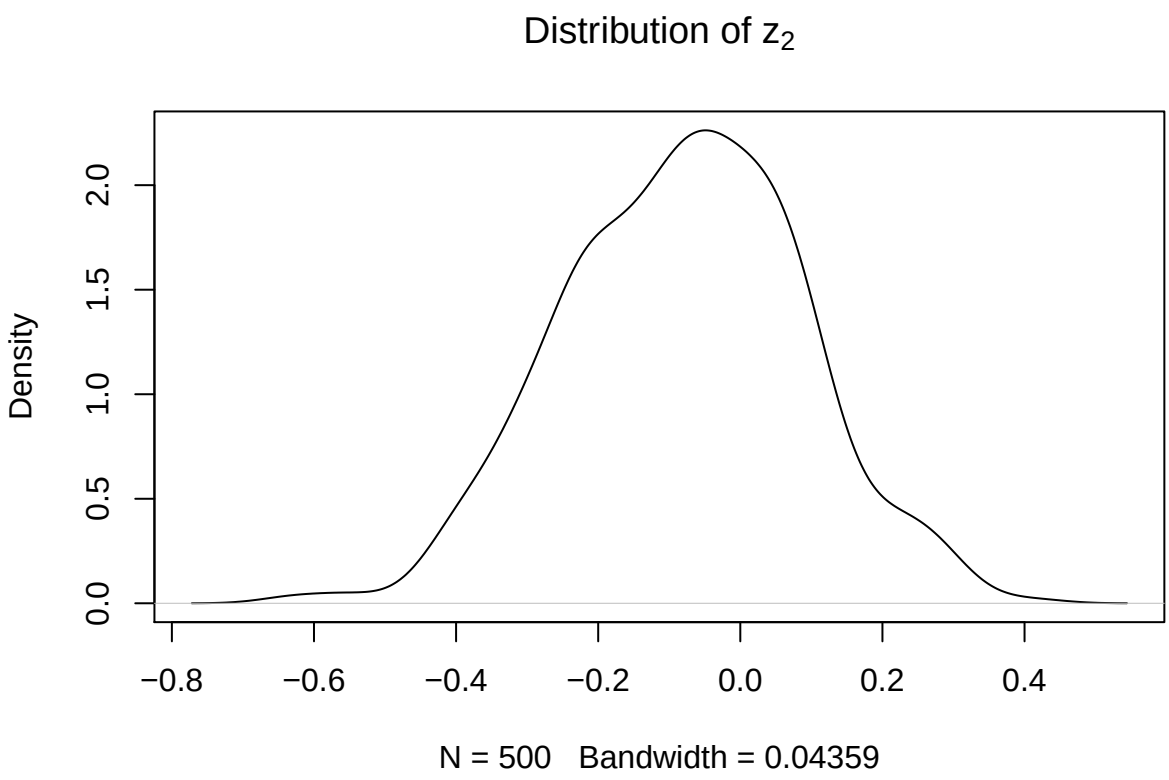
This distribution looks very similar to the distribution of $\hat{\mu_1}$. The mode is at zero which shows that most of the time, the estimate was approximately correct.

Estimate of μ_2

The true value of μ_2 is 2. As was mentioned earlier, the concern is that with the parameter values in this example causing a significant overlap between the two distributions, it may be difficult to detect the true parameter values. The plot below shows a distribution of the estimates $\hat{mu}_2$:



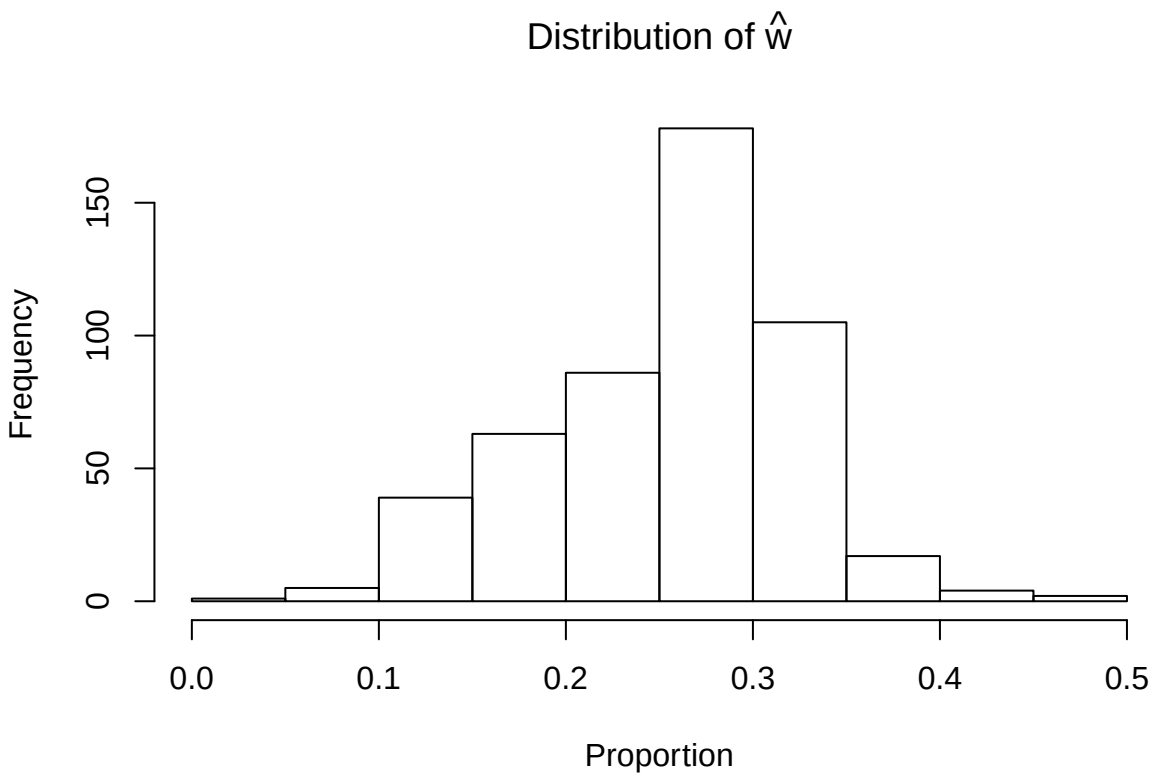
The algorithm does a fairly good job estimating μ_2. The estimates do not go below 1.4 because the other component of the distribution is in that direction.



Again, this distribution looks very similar to the distribution of $\hat{\mu_2}$.

Estimate of the proportion

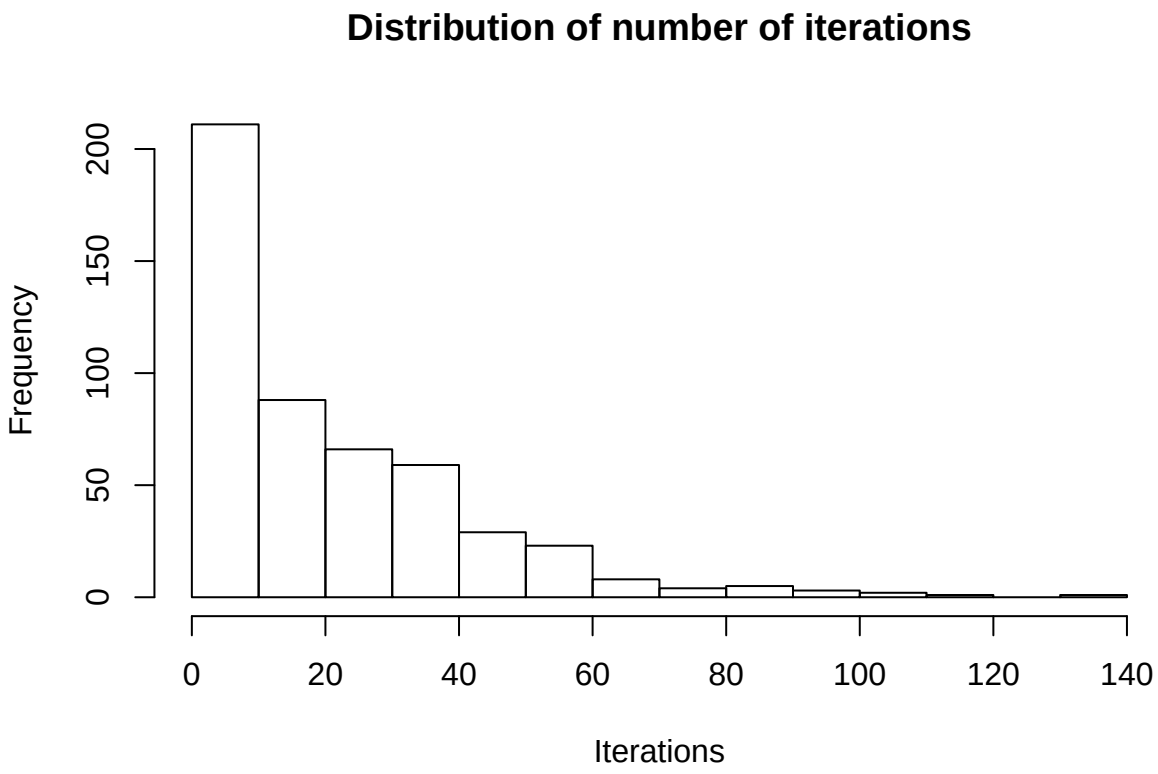
The distribution of the estimates for the proportions can be analyzed. The true value is 0.30 (and 0.70). Below is the distribution of the proportion estiamtes:



With the particular initial guesses that were used and the particular true values that were used, it seems the the algorithm tends to output a fairly good approximation of the true proportion. Since the first component has a less weight, it makes sense that this distribution has a heavier left tail. In some runs of the algorithm, it has a hard time recognizing that points from the first component in fact came from the first component.

Distribution of number of iterations

Below is a histogram of the number of iterations needed for the algorithm to converge in the 500 trials:



The majority of trials required less than 60 iterations in order to converge.

Conclusion

Repeating the algorithm 500 times, the average values of the estimates across all 500 trials were:

Value	Mean Estimate
$\overline{\hat{\mu_1}}$	-1.40389701
$\overline{\hat{\mu_2}}$	1.91890118
$\overline{\hat{\sigma_1}}$	1.74920379
$\overline{\hat{\sigma_2}}$	1.06406263
$\overline{\hat{w}}$	0.25533168
Iterations	22.286

Overall over 500 runs of the algorithm, the average performance was very satisfactory. Estimates for μ_1 tended to be underestimates while estimates for μ_2 tended to be close to the correct value of 2. There are a lot of questions that remain to be answered however. For example, what characteristics did the trials that took a long time to converge have? Did the trials that took a long time to converge have good estimates or bad estimates? It would also be expected that the closer the two mixture components are to each other, the worse the algorithm would perform, but this could be investigated by simulations to actually demonstrate.

Appendix

The code below was used to run the simulation:

```
# Keep track of z-scores and number of iterations until convergence
results <- data.frame(mu1 = rep(0, 500),
                      mu2 = rep(0, 500),
                      sigma1 = rep(0, 500),
                      sigma2 = rep(0, 500),
                      prop = rep(0, 500),
                      z1 = rep(0, 500),
                      z2 = rep(0, 500),
                      iterations = rep(0, 500))

# Run EM algorithm 500 times on same problem
for(i in 1:500){
  # Initial parameter values guesses
  mu1_hat <- -0.5
  mu2_hat <- 0.5
  sig1_hat <- 1
  sig2_hat <- 1
  weight <- 0.5

  # Calculate how many values will come from each component
  sample1_size <- rbinom(n = 1, size = sample_size, prob = proportion)
  sample2_size <- sample_size - sample1_size

  # Sample that many values from each respective normal distribution
  sample1 <- rnorm(n = sample1_size, mean = mu_1, sd = sigma_1)
  sample2 <- rnorm(n = sample2_size, mean = mu_2, sd = sigma_2)
  sample <- c(sample1, sample2)

  # number of iterations
  iterations <- 0

  while(TRUE){
    # increment iterations
    iterations <- iterations + 1

    # E step
    z_hat <- (weight * dnorm(sample, mu1_hat, sig1_hat)) /
      ((weight * dnorm(sample, mu1_hat, sig1_hat))
       + (1 - weight) * dnorm(sample, mu2_hat, sig2_hat))

    # M step
    m_mu1_hat <- sum(z_hat * sample) / sum(z_hat)
    m_mu2_hat <- sum((1 - z_hat) * sample) / sum(1 - z_hat)
    m_sig1_hat <- sqrt(sum(z_hat * (sample - mu1_hat)^2) / sum(z_hat))
    m_sig2_hat <- sqrt(sum((1 - z_hat) * (sample - mu2_hat)^2) / sum(1 - z_hat))
    m_weight <- sum(z_hat) / 100

    # Stopping condition
    if(abs(mu1_hat - m_mu1_hat) < 0.01){
      # store values and break loop
```

```
      results$mu1[i] <- m_mu1_hat
      results$mu2[i] <- m_mu2_hat
      results$sigma1[i] <- m_sig1_hat
      results$sigma2[i] <- m_sig2_hat
      results$prop[i] <- m_weight
      results$z1[i] <- (m_mu1_hat - mu_1) / sigma_1
      results$z2[i] <- (m_mu2_hat - mu_2) / sigma_2
      results$iterations[i] <- iterations
      break
    } else{ # update values and continue loop
      mu1_hat <- m_mu1_hat; mu2_hat <- m_mu2_hat
      sig1_hat <- m_sig1_hat; sig2_hat <- m_sig2_hat
      weight <- m_weight
    }
  }
}
```