MLE of Mixed Normal Distribution

Jiarong Feng*
Department of Statistics, University of California, Irvine
and
Qi Wang†
Department of Statistics, University of California, Irvine

October 16, 2020

Abstract

This algorithm allow you to use Newton's Method to calculate the MLE of one of the means of the two normal distirbuiton, given the percentage θ, standard error σ_1, σ_2, data set Y, and one of the mean $\mu_i, i = 1, 2$. We are using Newton's Method, after calculating the score function of the likelihood and get the root of score function to check where will the iteration converge. After we find the domain of convergence, we also need to check the second derivative of the root, to make sure that we are getting the maximum value but not something else of the likelihood function.

Keywords: Mixed Normal Distribution, MLE, Newton's Method, Score Functionn

*jiarongf1@uci.edu
†qwang18@uci.edu

1 Introduction to Mixed Normal Distribution

Mixed normal distribution is made up with two normal distribution, and with some percent of each of them.

The pdf of this new distribution is like:

$$p(y_i|\theta) = \theta\phi(y_i; \mu_1, \sigma_1^2) + (1-\theta)\phi(y_i; \mu_2, \sigma_2^2)$$

And here $\phi(y_i; \mu_i, \sigma_i^2)$ is the normal pdf, which means that:

$$\phi(y_i; \mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(y_i - \mu_i)^2}{-2\sigma_i^2})$$

Our goal is to calculate the MLE of μ_1 given the other μ_2, θ, σ_1 *and* σ_2

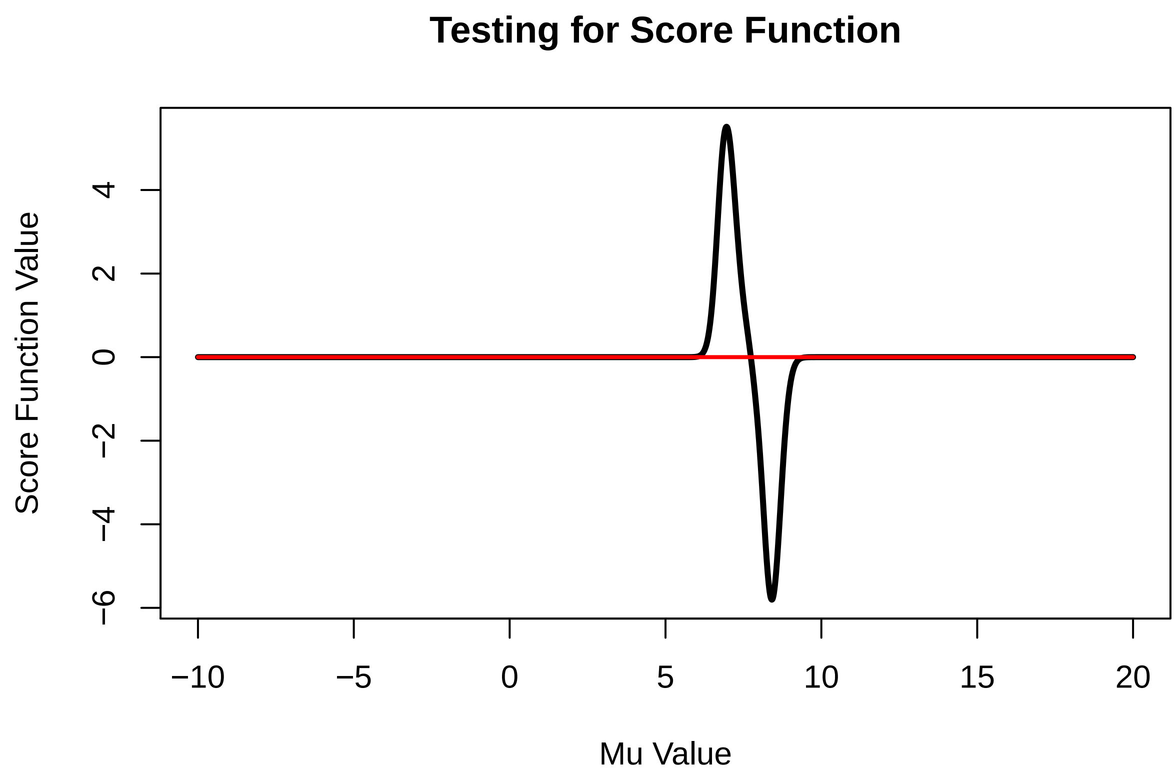
2 Code

```
g <- function(y, theta, mu_1, mu_2, sigma_1, sigma_2){

  score <- rep(NA,length(y))

  for(i in 1:length(y)){

    score[i] <- ( (theta*(y[i] - mu_1)/sigma_1^2)*dnorm(y[i], mu_1, sigma_1 ) ) /
      ( (theta* dnorm(y[i], mu_1, sigma_1 )) + (1-theta)*dnorm(y[i], mu_2, sigma_2) )

  }

  return(sum(score))
}

deriv_g <- function(y, theta, mu_1 , mu_2, sigma_1, sigma_2, h = 1e-2){

   derivative <- (g(y, theta, mu_1+h , mu_2, sigma_1, sigma_2)-
```

```r
                     g(y, theta, mu_1-h , mu_2, sigma_1, sigma_2))/(2*h)

   return(derivative)
}

mixture_normal.mle <- function(y, theta, mu_1, mu_2, sigma_1, sigma_2,
                                max.iter = 1000, small = 1e-3, h = 1e-2){

  for(iter in 1:max.iter){
    mu_1.old <- mu_1
    mu_1 <- mu_1.old - g(y,theta,mu_1.old, mu_2, sigma_1, sigma_2)/
      deriv_g(y,theta,mu_1.old, mu_2, sigma_1, sigma_2, h)

    if(abs(mu_1.old - mu_1) <= small)
      break
  }

  out <- list(mu_1, iter, iter != max.iter)
  names(out) <- c("MLE", "iteration.count", "converge")
  out
}


y <- c(8.1, 8.2, 8.1, 8.2, 8.2, 7.4, 7.3, 7.4, 8.1,
       8.1, 7.9, 7.8, 8.2, 7.9, 7.9, 8.1, 8.1)
```

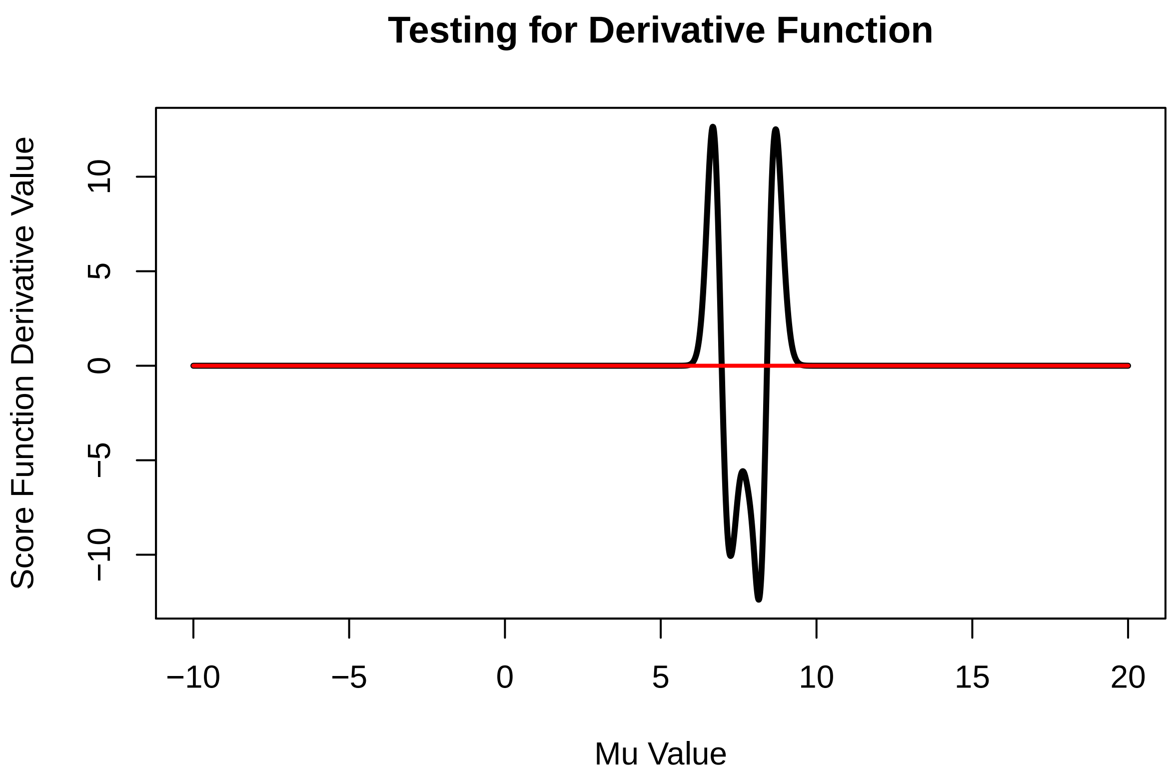
Here we used a dataset y to verify whether we have the right MLE, since Newton's method is unstable, sensitive to the initial point.

3 Test whether score function is wrong



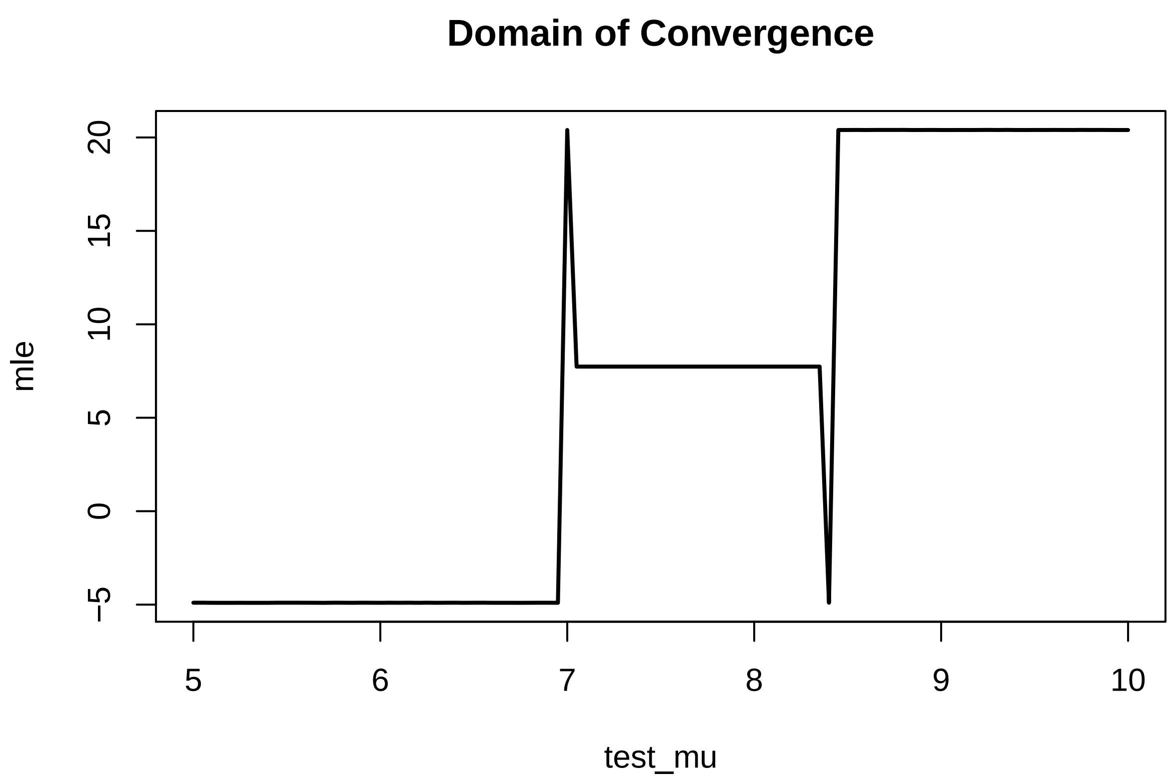
It seems that there are many many zero points. Don't forget we are using the zero point of score function to calculate the MLE!

4 Test whether derivative of score function is wrong



According to the definition of second derivative, and first derivative, the function has the maximum value when the first derivative is near 8. Because the LLH function value has been increasing until that point. Also, the second derivative is negative according to picture 2. So the Derivative function is also right. **But we notice that, only when mu_1 is between 5 and 10, will the second derivative not equal to 0.**

5 Chosing Initialization Values and Calculate Domain of Convergence



It seems right since there is a stable area in -5,8 and 20. So the domain of convergence for them are obvious. However, there are many special points around 7 and 8.35. Why are they there? I set the sequence break to be 0.05, but I need to dig more into them to check what happened as test_mu changed from 6.95 to 7.

6 Special Points

Around 7:

When I type in the code as follows:

test_mu_7 = seq(6.95,7,0.001)

mle_7 = vector()

for (i in 1:length(test_mu_7)){

mle_7[i] = mixture_normal.mle(y, theta = 0.2, mu_1 = test_mu_7[i], mu_2 = 8.0, sigma_1 = sqrt(0.1), sigma_2 = sqrt(0.1))$“MLE”

}

plot(x = test_mu_7, y = mle_7, lwd = 2, type = ‘l’)

Error in if (abs(mu_1.old - mu_1) <= small) break :

TRUE/FALSE

This might be caused by the special case near 7, leading to the derivatives hard to calculate! Please check the score function, which is the derivative for MLE, around 7, it seems that the function is not differenciable. Similar around 8.

7 Conclusion

1. Change and adjust the precision(you named it as “small”), and h(which is used to calculate second derivative for log-likelihood function).

2. Choose initialization value of μ_1 a little far from the place that is not differenciable.