Project 3: Analysis of Fetal Mortality Data

Yi Han

March 9,2020

1 Summary

Brooks et al.(1997)[1] presented six overdispersed dataset recording fetal mortality in mouse litters and fitted eight different models to predict the data. Comparing with the standard approach, describing the data by means of a beta-binomial model, the author showed that the mixure models can offer a better fit of the dataset in the sense of producing a larger maximum likelihood because they can explain outliers to some extent.

In their paper, they first use simulations to test goodness-of-fit for the maximum log-likelihood obtained by the beta-binomial model. They found that the real data value of the maximum log-likelihood is in the middle of simulated maximum log-likelihood, which suggest the beta-binomial model provides a resonable fit of the dataset. However, because of the existence of the outliers, they proposed some different new model, finite mixture model, to fit the data. They build eight different models: standard models including binomial, correlated binomial, beta binomial and beta-correlated binomial; mixture models including two-element binomial mixture, three-element binomial mixture, nonparametric mixtures of binomial distributions and a mixture of a beta-binomial with a binomial model.

From fitting the eight models by maximum likelihood, they obtain the maximized log-likelihood functions. The results showed that the nonparametric mixtures of binomial distributions has the largest maximum likelihood value. Besides, the author mentioned that five of six dataset can be improved by using the mixture models compared with the original model in the sense of getting larger maximum likelihood values. Simulation studies are followed to guage goodness-of-fit so that they can compare the estimates performance of the models. The author mentioned that it is quite reasonable to use the beat-binomial model when given the litter data with overdispersion but since there are some outliers which represent the high morality of some litters, it is straight forward to think a mixture effect to represent such effct in in the dataset. So when the effect of random model were better than the original one, it is reliable to use the random effect model than the original one. The Monte Carlo tests are as expected that favor mixture models than single binomial models. One of them is that the data are contaminated by the a small fraction of the outliers. In this case, one important part is to obtain the mixing parameter γ. The authors showed that one can obtain confidence intervals of γ from the profile log-likelihood. The mixture models also should be investigated in small dataset especially when a biological basis can be found for the components of the mixture variable.

2 Principle of the Monte Carlo EM algorithm

Maximum lilelihood estimator(MLE) is widely used in estimating the parameters of a statistical model, given observations. Let the data given is: $X = (X_1, X_2, \ldots, X_n)$ be a sample with density $f(x;\theta)$, the procedure for computing the MLE is the following:

1.Under the assumption of the model, get the log-likelihood function of X:

$$\hat{l}(\theta; X) = logf(X;\theta) = \sum_{i=1}^{n} logf(X;\theta)$$

2. Find $\hat{\theta}$ to maximize the average log-likelihood function

$$\hat{\theta} = argmax\hat{l}(\theta; X)$$

However, things are not always so straight forward when "missing data" occurs or there exists some random effects that can not be observed directly in the model. For example, $X = (Y, Z)$, where $Y = y$ is the observed data, Z is the unobserved part(missing data or random effects). Then the MLE of θ is the value of $\hat{\theta}$ which maximizes the log-likelihood function for the observed data y:

$$\hat{l}(\theta; y) = log \int f(y, z;\theta)u(dz)$$

Computing the MLE of above maybe difficult because it requires to compute the integral above. EM algorithm can be used even if the integral of the above is intractable. Define $Q(\theta, \hat{\theta})$ to be the expected value of the "complete data" log-likelihood, conditionally on the observed data y. Namely

$$Q(\theta, \tilde{\theta}) = E_{Z|y;\tilde{\theta}}\{logf(y, z;\theta)\} = \int [logf(y, z;\theta)]h(z|y;\tilde{\theta})dz$$

where $h(z|y;\tilde{\theta})$ is the conditional distribution of Z given $Y = y$ and current θ value $\tilde{\theta}$.

So the EM algorithm can be described like these two steps:

1.E-step: Calculate the conditional expectation of the log likelihood function of θ, with respect to the current conditional distribution of Z given X and the current estimates of the paremeter $\tilde{\theta}$:

$$Q(\theta, \tilde{\theta}) = E_{Z|y;\tilde{\theta}}\{logf(y, z;\theta)\}$$

2.M-step: Find the paremeters that maximize this quantity:

$$\theta_{new} = \arg\max_{\theta} Q(\theta|\tilde{\theta})$$

Then substitude the $\tilde{\theta}$ by θ_{new} and then do the two steps again. Doing such iteration until θ to be convergent. However, here it is difficult to calculate the E-step because here $h(z|y;\tilde{\theta})$ is intractable. So here we use the MH algorithm to sample from the distribution $h(z|y;\tilde{\theta})$.

3 Analysis using the Maximum Likelihood

The main idea for MH algorithm is that the E-step can be approximated by Monte Carlo integration, using a random sample $z^{(1)}, z^{(2)}, \ldots, z^{(m)}$ from the target distribution

$$h(z|y;\tilde{\theta}) \propto f(y|z;\theta^{(t)})h(z;\theta^{(t)})$$

.

Then a Monte Carlo approximation to $Q(\theta, \hat{\theta})$ is given by:

$$Q_m(\theta, \theta^{(t)}) = \frac{1}{m}\sum_{k=1}^{m} logf(y, z^{(k)};\theta)$$

How to sample from the target distribution is also a question that needs to be investigated. If direct simulation from $h(z|y;\tilde{\theta})$ is difficult, one might resort to ta Markov chain Monte Carlo(MCMC) method such as the Metropolis-Hasting algorithm. Here we present the procedure for sampling using the Metropolis-Hasting algorithm briefly.

First, decide a jumping distribution function $q(x, y)$ such that $q(x, .)$ is a pdf for every x. Given the current value $z^{(k)} = z_0$, a candidate value for $z^{(k+1)}$, say z_1, is sampled from jumping distribution $q(z_0, .)$. A decision is made on whether or not "jump"(accept the candidate value) based on a acceptance ratio:

$$\alpha = min(\frac{f(z_1)q(z_1, z_0)}{f(z_0)q(z_0, z_1)}, 1)$$

where f is the target distribution of sampling. Then $z^{(k+1)} = z_1$ (jumping to the canidate) with probability α; $z^{(k+1)} = z^{(k)}$ (staying at the current value) with probability $1 - \alpha$.

Note that when the target distribution is the above $h(z|y;\tilde{\theta})$ in MCEM, choosing the marginal distribution $h(z;\theta^{(t)})$ as the jumping distribution $q(z_0, .)$ will lead to the acceptance rate have a simpler form:

$$\frac{f(z_1)q(z_1, z_0)}{f(z_0)q(z_0, z_1)} = \frac{h(z_1|y;\theta^{(t)})q(z_1, z_0)}{h(z_0|y;\theta^{(t)})q(z_0, z_1)} = \frac{f(y|z_1;\theta^{(t)})h(z_1;\theta^{(t)})}{f(y|z_0;\theta^{(t)})h(z_0;\theta^{(t)})} \cdot \frac{q(z_1, z_0)}{q(z_0, z_1)} = \frac{f(y|z_1;\theta^{(t)})}{f(y|z_0;\theta^{(t)})}$$

Hereafter, we will use this jumping distribution in Metropolis-Hasting algorithm. This algorithm actually generates a Markov chain, and the onvergence distribution of the chain is the target distribution. Hence, once the convergence is reached, one can throws out the "burn-in", i.e., the first T samples $z^{(1)}, z^{(2)}, \ldots, z^{(T)}$, and only keeps the rest of chains, $z^{(T+1)}, \ldots, z^{(T+m)}$ for distribution related computations.

To be more specific in this paper, we will analyze HS2 data of Brooks et al.(1997)[1] using Monte Carlo EM algorithm. This dataset records the number of dead implants in 1328 litters of mice from untreated experimental animals. Considering every implant as an individual response, all the responses can be denoted as $y_{ij}, i = 1, \ldots, 1328, j = 1, \ldots, n_i$ where n_i is the number of implants of the ith litter. Let $y_{ij} = 1$ if the $j-$th implant in $i-$th

litter is dead; $y_{ij} = 0$ otherwise. The total number of responses is $N = \sum_{i=1}^{1328} n_i = 10533$. As what Jiang(2007)[3] did, we also consider a GLMM model. Suppose that, given the random effects $\alpha_1, \alpha_2, \ldots, \alpha_I$, the binary responses $y_{ij}, 1 \leq i \leq I = 1328, 1 \leq j \leq n_i$ are conditionally independent such that

$$logit\{P(y_{ij} = 1|\alpha)\} = u + \alpha_i$$

where u is an unknown parameter. Futhermore, suppose α_i's are independent and disstributed as $N(0, \sigma^2)$. Note that here $I = 1328$ so that α_i can be explained as the random effect related to the $i-$th litter. Our task is to find the MLE of u and σ. Let the unobserved data Z in (1) equals to $\alpha, \theta = (u, \sigma)$, then we can use the Monte Carlo algorithm introduce in section 2. The Q-function is

$$Q(\theta, \tilde{\theta}) = E\{logf(y, \alpha; \theta)|y; \tilde{\theta}\} = E\{logf(y|\alpha; u)|y; \tilde{\theta}\} + E\{logf(\alpha; \sigma)|y; \tilde{\theta}\}$$

Thus we can optimize over u and σ seperately. In other to evaluate the two integrals above using Monte Carlo approach, we need to get samples of α from the target distribution $h(\alpha|y)$, so the Metropolis-Hasting algorithm is used in this step. Let the current parameter value be $\theta^{(t)}$, then the whole algorithm for one iteration is as following:

1.E-step: First use Metropolis-Hasting method to get a random sample, $\alpha^{(t,1)}, \ldots, \alpha^{(t,m)}$, from the conditional density $h(\alpha|y; \theta^{(t)})$. Then evaluate (6) by

$$Q_1(u, \theta^{(t)}) = \frac{1}{m} \sum_{k=1}^{m} logf(y|\alpha^{(t,k)}; u)$$

where $f(y|\alpha; u) = \prod_{i=1}^{I} \prod_{j=1}^{n_i} \frac{e^{(u+\alpha_i)y_{ij}}}{(1+e^{(u+\alpha_i)})^{y_{ij}}} \cdot \frac{1}{(1+e^{u+\alpha_i})^{(1-y_{ij})}}$

$$Q_2(\sigma, \theta^{(t)}) = \frac{1}{m} \sum_{k=1}^{m} logh(\alpha^{(t,k)}|\sigma)$$

where $f(\alpha|\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2\sigma^2} \cdot \sum_{i=1}^{n} \alpha_i^2}$

: Maximize $Q_1(u, \theta^{(t)})$ and $Q_2(\sigma, \theta^{(t)})$ and get the updates by

$$u^{(t+1)} = argmaxQ_1(u, \theta^{(t)})$$

,

$$\sigma^{(t+1)} = argmaxQ_2(\sigma, \theta^{(t)})$$

, This procedure is iterated until it converges. The starting value can be choosen randomly here.

Then we implement MCEM algorithm for 300 times and plot $u^t, \sigma^t, t = 1, \ldots, 300$. Then we can see that u converges to the interval $[-2.28, -2.275]$, σ converges to interval $[0.67, 0.68]$.

From the plot we can see that after 30 times' iteration, the result converges, so I also draw the result after it converges in the ideal iterval of u and σ.

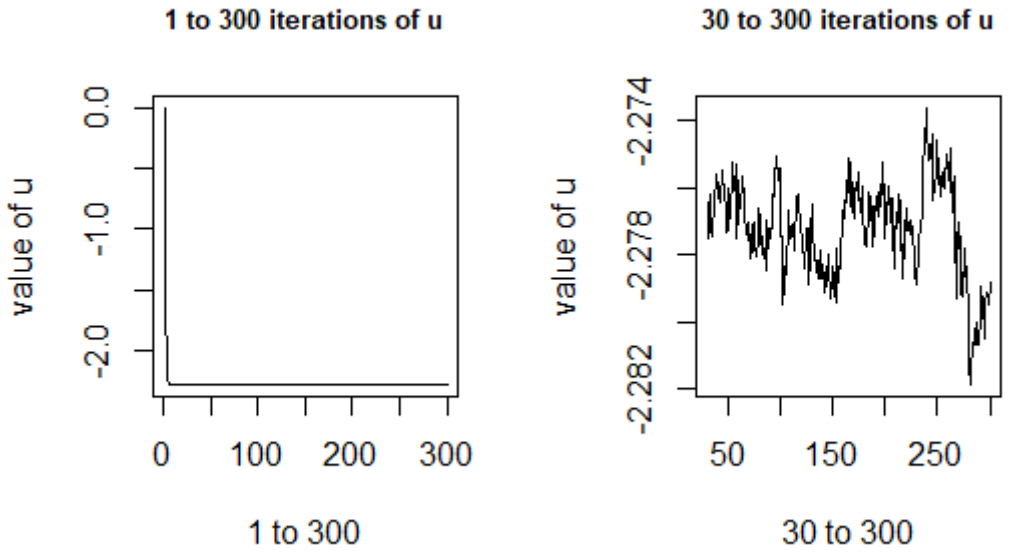

Figure 1: The result of u using MCEM

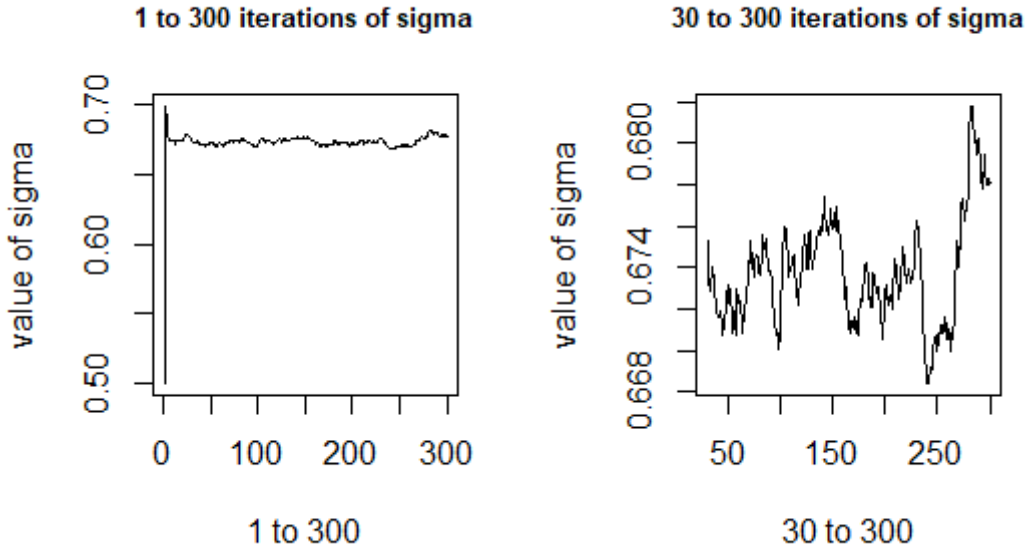

Figure 2: The result of σ using MCEM

4 Standard error estimation

According to the Louis(1982)[2], although the complete log likelihood function of compelete data is intractable, there is a way to the so called observed information matrix which is very close to the real on. The MLE of θ is denoted as $\hat{\theta}$, then the observed information matrix can be calculated as:

$$I(\theta, y) = -E_{\alpha|y}\{\frac{\partial^2 l(\theta; y, \alpha)}{\partial\theta\partial\theta'}\}|_{\theta=\hat{\theta}} - E_{\alpha|y}\{\frac{\partial l(\theta; y, \alpha)}{\partial\theta} \cdot [\frac{\partial l(\theta; y, \alpha)}{\partial\theta}]^T\}|_{\theta=\hat{\theta}}$$

where the expectation are with respect to $f(\alpha|y)$.

Here, I use the value of α and σ in the last iteration of MCEM algorithm where $\hat{\theta} = (-2.278, 0.678)$. Here we get the result:

$$I(\hat{\theta})^{-1/2} = \begin{pmatrix} 0.0212 & -0.0009 \\ -0.0009 & 0.0119 \end{pmatrix}$$

References

[1] Brooks S P, Morgan B J T, Ridout M S, et al. Finite mixture models for proportions[J]. Biometrics, 1997: 1097-1115.

[2] Jiang J. Linear and generalized linear mixed models and their applications[M]. Springer Science & Business Media, 2007.

[3] Louis T A. Finding the observed information matrix when using the EM algorithm[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1982, 44(2): 226-233.

```r
library("lme4")

nlist = list()
nlist[[1]] = c(15,1)
nlist[[2]] = c(6,1,2)
nlist[[3]] = c(6,6)
nlist[[4]] = c(7,2,3,NA,2)
nlist[[5]] = c(16,9,3,3,1)
nlist[[6]] = c(57,38,17,2,2)
nlist[[7]] = c(119,81,45,6,1,NA,NA,1)
nlist[[8]] = c(173,118,57,16,3,NA,NA,NA,1)
nlist[[9]] = c(136,103,50,13,6,1,1)
nlist[[10]] = c(54,51,32,5,1,NA,NA,NA,NA,1)
nlist[[11]] = c(13,15,12,3,1)
nlist[[12]] = c(NA,4,3,1)
nlist[[13]] = c(NA,NA,1,NA,NA,NA,NA,1)

myobs = vector()
count = 0
for (k in 1:13) {
  for (l in 1:length(nlist[[k]])) {
    if(!is.na(nlist[[k]][l]))
      for (m in 1:nlist[[k]][l]) {
        count = count + 1
        myobs = rbind(myobs,cbind(c(rep(0,k-l+1),rep(1,l-1)),count))
      }
  }
}
myobs = data.frame(y=myobs[,1],litter=factor(myobs[,2]))

model1 = glmer(y~1+(1|litter),data = myobs,family = "binomial")
summary(model1)
initial_mu = fixef(model1)
initial_sigma = sqrt(unlist(VarCorr(model1)))
#####################################################################
I = nlevels(myobs$litter)
y_i = split(myobs$y,myobs$litter)
sum_y_i = sapply(y_i, sum)
n_i = sapply(y_i, length)
m = 10000
burn_out = 500

#################################################3
y_likelihood = function(mu,alpha){
```

```
  return(sum_y_i%*%(mu+alpha)-n_i%*%log(1+exp(mu+alpha)))
}

alpha_likelihood = function(alpha,sigma){
  return(sum(dnorm(alpha,mean = 0,sigma,log = T)))
}

#compute the acceptance rate
r = function(alpha0,alpha1,mu){
  p = sum_y_i*(alpha1-alpha0)
  - n_i*log((1+exp(mu+alpha1))/(1+exp(mu+alpha0)))
  return(pmin(exp(p),1))
}

#sample from the target distribution: f(alpha|y,mu)
metropoli = function(mu,sigma){
  iterations = m + burn_out
  initial_alpha = rnorm(I,0,sigma)
  alpha = matrix(0,I,iterations)
  alpha[,1] = initial_alpha
  for (i in 2:iterations) {
    alpha_candidate = rnorm(I,0,sigma)
    u = runif(I)
    r = r(alpha[,i-1],alpha_candidate,mu)
    jump = which(u<r)
    length(jump)
    alpha[jump,i] = alpha_candidate[jump]
    alpha[-jump,i] = alpha[-jump,i-1]
  }
  burn_out_alpha = alpha[,-(1:burn_out)]
  return(burn_out_alpha)
}

set.seed(1)
z0 = metropoli(-2.99,0.69)
write.table(z0,"z0.txt")
plot(z0[1,500:10000],type="l")

##########################optimize
optimize_mu = function(alpha,mu){
  return(-mean(apply(alpha, 2, y_likelihood,mu=mu)))
}

optimize_sigma = function(alpha,sigma){
```

```
  return(-alpha_likelihood(alpha,sigma))
}

#optim(-2.29,optimize_mu,alpha=z0,method="BFGS")
#######
MCEM = function(mu_initial,sigma_initial,times){
  mu = rep(0,times)
  mu[1] = mu_initial
  sigma = rep(0,times)
  sigma[1] = sigma_initial
  for (i in 2:times) {
    alpha = metropoli(mu[i-1],sigma[i-1])
    mu[i] = optim(mu[i-1],optimize_mu,alpha=alpha,method="L-BFGS-B"
                  ,lower=-3,upper=-1)$par
    sigma[i] = optim(sigma[i-1],optimize_sigma,alpha=alpha,
                     method="L-BFGS-B",lower=0.3,upper=0.8)$par
    dmu = abs(mu[i]-mu[i-1])
    dsigma = abs(sigma[i]-sigma[i-1])
    cat("mu is:",mu[i],"df is:",dmu,"Iteration is:",i,"\n")
    cat("sigma is:",sigma[i],"df is:",dsigma,"\n")
  }
  return(list(mu,sigma))
}

set.seed(1)
result = MCEM(0,0.5,300)
names(result)=c("mu","sigma")
write.table(result,"result.txt")
par(mfrow=c(1,2))
plot(c(1:300),result$mu,type="l",main = "1 to 300 iterations of u",
     xlab="1 to 300",ylab = "value of u",cex.main=0.8)
plot(c(30:300),result$mu[30:300],type = "l",
     main = "30 to 300 iterations of u",
    xlab="30 to 300",ylab = "value of u",cex.main=0.8)

plot(c(1:300),result$sigma,type="l",
     main = "1 to 300 iterations of sigma",
     xlab="1 to 300",ylab = "value of sigma",cex.main=0.8)
plot(c(30:300),result$sigma[30:300],type = "l",
     main = "30 to 300 iterations of sigma",
     xlab="30 to 300",ylab = "value of sigma",cex.main=0.8)

mu = result$mu[300]
sigma = result$sigma[300]
```

```
alpha = z0
s1 = sum(sum_y_i)-sum(n_i%*% ((1/(1+exp(mu+alpha)))*exp(mu+alpha)) )
s2 = -10533/sigma+sum(alpha^2/sigma^3)
s = as.matrix(c(s1,s2),ncol=1)
B1 = sum(n_i %*% (exp(mu+alpha)/(1+exp(mu+alpha))^2))
B4 = 10533/sigma^2-3/(sigma^4)*sum(alpha^2)
B = matrix(c(B1,0,0,B4),ncol=2)
I = -B + s %*% t(s)
(solve(I))^(1/2)*100
```