Fall 2016 BIOSTAT 815 Problem Set #2

Submission Guide:

- Due is Monday October 10th 11:59pm.
- You will need to submit the following items via Canvas and google doc
  1. A zip (or .tar.gz) file containing the R package R, and your report (pdf, doc, or docx format).
  2. The copy of R and C++ code used in the package also need to be available at google doc (under your umich.edu account) for grading.
  3. For this homework, providing a comprehensive explanation of the results is as important as writing the code correctly and efficiently.
  4. Make the google doc document editable by anyone (within University of Michigan) who has the link, and include the link in your report.

Problem 1. Gaussian Mixture E-M and Gibbs Sampler

As we leanred from the class, an E-M algorithm based on a k-component Gaussian mixture model repeats the following two steps until convergence.

E-step For $i = 1, \ldots, n$, $j = 1, \ldots, k$, and $t = 0, 1, \ldots$, compute

$$w_{ij}^{(t)} = \Pr(z_i^{(t)} = j | \theta^{(t)}) = \frac{\pi_j^{(t)} \mathcal{N}(x_i | \mu_j^{(t)}, (\sigma_j^2)^{(t)})}{\sum_{m=1}^{k} \left[\pi_m^{(t)} \mathcal{N}(x_i | \mu_m^{(t)}, (\sigma_m^2)^{(t)}) \right]}$$

where $\mathcal{N}(\cdot)$ is a pdf of normal distribution.

M-step For $j = 1, \ldots, k$, update following parameters

$$\begin{aligned}
\pi_j^{(t+1)} &= \frac{1}{n} \sum_{i=1}^{n} w_{ij}^{(t)} \\
\mu_j^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} x_i}{\sum_{i=1}^{n} w_{ij}^{(t)}} \\
(\sigma_j^2)^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} (x_i - \mu_j^{(t)})^2}{\sum_{i=1}^{n} w_{ij}^{(t)}}
\end{aligned}$$

Similarly, a Gibbs sampler based on a k-component Gaussian mixture model samples $z_i \in \{1, \ldots, k\}$ for a sequentially or randomly chosen i based the following conditional distribution:

$$\Pr(z_i^{(t+1)} = c | \mathbf{z}^{(t)}, \mathbf{x}) = \frac{\pi_c^{(t)} \mathcal{N}(x_i | \mu_c^{(t)}, (\sigma_c^2)^{(t)})}{\sum_{j=1}^{k} \pi_j^{(t)} \mathcal{N}(x_i | \mu_j^{(t)}, (\sigma_j^2)^{(t)})}$$

where $c \in \{1, \ldots, k\}$, $z_m^{(t+1)} = z_m^{(t)}$ when $m \neq i$, and $\mathcal{N}(\cdot)$ is a pdf of normal distribution.
$(\pi_1^{(t)}, \ldots, \pi_k^{(t)})$, $(\mu_1^{(t)}, \ldots, \mu_k^{(t)})$, $(\sigma_1^{(t)}, \ldots, \sigma_k^{(t)})$ are defined as

$$\begin{aligned}
\pi_j^{(t)} &= \frac{\sum_{m \neq i} I(z_m^{(t)} = j)}{n - 1} \\
\mu_j^{(t)} &= \frac{\sum_{m \neq i} I(z_m^{(t)} = j) x_w}{\sum_{m \neq i} I(z_m^{(t)} = j)} \\
(\sigma^2)_j^{(t)} &= \frac{\sum_{m \neq i} I(z_m^{(t)} = j)(x_w - \mu_j^{(t)})^2}{\sum_{m \neq i} I(z_m^{(t)} = j)}
\end{aligned}$$

(a) Implement an R package that implements E-M algorithm and Gibbs sampler efficiently using C++. Here are detailed requirements.

- You need to implement two functions `rcppEM()` end `rcppGibbs()` following the specifications described below.
- The E-M algorithm and Gibbs sampler should follow the same scheme to the R code presented in the lecture, except that it should be implemented in a C++ function.
- The time complexity of each Gibbs sampler iteration should be constant to the sample size.
- You need to create a R package named `[your-uniqname]RcppGibbsEM`. Your R package should export the following five functions.
  (a) `rcppEM()` to perform Gaussian Mixture E-M algorithm (C++)
  (b) `rcppGibbs()` to perform Gaussian Mixture Gibbs sampling (C++)
  (c) `simul.gmm.params()` to heuristically choose parameters for Gaussian Mixture (R)
  (d) `simul.gmm.k()` to randomly sample data from a specific Gaussian Mixture model (R)
  (e) `llk.gmm.k()` to compute log-likelihood based on Gaussian Mixture model (R)
- The resulting R package should be comprehensive, in terms of the ability to install by others (from the .zip or .tar.gz file), and documentation (brief and clear), correctness, etc.

For guidelines of creating an R package using `Rcpp`, please refer to `https://cran.r-project.org/web/packages/Rcpp/vignettes/Rcpp-package.pdf`.

Your C++ functions should conform to the following skeleton

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
List rcppEM(NumericVector x, int k, int maxIter = 1000) {
  //////////////////////////////////////////////////
  // Fill in the function to implement E-M algorithms
  //
  // For simplicity, you don't need to check convergence
  // and repeat E-M steps [maxIter] times.
  //
  // pis, mus, sds should be a length k vector
  // and llk should be a single double value
  //////////////////////////////////////////////////

  return List::create(Named("pis") = pis, Named("mus") = mus, Named("sds") = sds, Named("llk") = llk);
}

// [[Rcpp::export]]
List rcppGibbs(NumericVector x, int k, int burnIn = 1000000, int thinInterval = 10000,
               int nIter = 10000000, int minCount = 5) {

  //////////////////////////////////////////////////
  // Fill in the function to imlement Gibbs sampler
  //
  // Similar to the lecture notes,
  // pis, mus, sds should be m * k matrix
  // llks should be length m vector
  // where m = ( nIter - burnIn ) / thinInterval
  //
  // Make sure that each iteration has constant
  // time complexity
  //////////////////////////////////////////////////

  return List::create(Named("pis") = pis, Named("mus") = mus, Named("sds") = sds,
```

```
                    Named("llks") = llks, Named("burn_in") = burnIn,
                    Named("thin_interval") = thinInterval);
}
```

Your R code should contain the following three functions. These functions should be exported in the R package so that users can call these functions.

```r
sample.gmm.params <- function(k) {
    pis <- rnorm(k)
    pis <- 1/(1+exp(0-pis))
    pis <- pis/sum(pis)

    mus <- cumsum(rexp(k)+2)
    mus <- mus - mean(mus)

    sds <- sqrt(rexp(k))

    return( list(pis=pis, mus=mus, sds=sds) )
}

simul.gmm.k <- function(n, pis, mus, sds) {
    z <- apply(rmultinom(n, 1, pis),2,which.max)
    x <- rnorm(n, mus[z], sds[z])
    return(list(x=x,z=z))
}

llk.gmm.k <- function(x, pis, mus, sds) {
    n <- length(x)
    k <- length(pis)
    return(sum(log(rowSums(dnorm(matrix(x,n,k), matrix(mus,n,k,byrow=TRUE), matrix(sds,n,k,byrow=TRUE))
                           * matrix(pis,n,k,byrow=TRUE)))))
}
```

(b) Use the R package to evaluate these algorithms using the

```
install.packages('/path/to/my/package.tar.gz', repo=NULL, type=source)

library(uniqnameRcppGibbsEM)  ## change uniqname to yours
library(ggplot2)

## This function calculates difference between
## two pdfs (one from model, one from estimated)
integrate.diff.gmm <- function(model, estim) {
    ngrids <- 10000
    zmax  <- 100
    grids <- ((0:ngrids)/ngrids-0.5)*zmax*2

    pdf.model <- rep(0,ngrids+1)
    pdf.estim <- rep(0,ngrids+1)

    for(i in 1:length(model$pis)) {
        pdf.model = pdf.model + dnorm(grids, model$mus[i], model$sds[i]) * model$pis[i]
    }

    for(i in 1:length(estim$pis)) {
        pdf.estim = pdf.estim + dnorm(grids, estim$mus[i], estim$sds[i]) * estim$pis[i]
    }

    return(sum(abs(pdf.model-pdf.estim))*(2*zmax/ngrids))
}

## This function evaluates E-M algorithm and Gibbs sampler
## by comparing with the original model
evaluate.gs.em <- function(model, data, d) {
    k <- length(model$pis)
    gs.time <- system.time( gs.sample <- rcppGibbs(data, d) )[3]
    em.time <- system.time( em.est <- rcppEM(data, d) )[3]
    gs.mean <- list( pis = colMeans(gs.sample$pis),
                     mus = colMeans(gs.sample$mus),
                     sds = colMeans(gs.sample$sds) )
    imax <- which.max(gs.sample$llks)
    gs.mle <- list( pis = gs.sample$pis[imax,],
                    mus = gs.sample$mus[imax,],
                    sds = gs.sample$sds[imax,] )

    llk0 <- llk.gmm.k(data, model$pis, model$mus, model$sds)

    return( list(time=c(em.time,gs.time,gs.time),
                 pdf.diff=c(integrate.diff.gmm(model, em.est),
                            integrate.diff.gmm(model, gs.mean),
                            integrate.diff.gmm(model, gs.mle)),
                 llk.diff=c( llk.gmm.k(data, em.est$pis, em.est$mus, em.est$sds) - llk0,
                            llk.gmm.k(data, gs.mean$pis, gs.mean$mus, gs.mean$sds) - llk0,
                            llk.gmm.k(data, gs.mle$pis, gs.mle$mus, gs.mle$sds) - llk0 ),
                 model.dim = k,
                 est.dim = d ) )
}

run.gibbs.eval <- function() {
    ks <- c(2,5,10);

    for(k in ks) {
      df <- data.frame("dim.model"=NULL,"dim.est"=NULL,"method"=NULL,
```

```
                    "pdf.diff"=NULL,"llk.diff"=NULL,"time.sec"=NULL)
        for(d in ks) {
            print(c(k,d))
            for(j in 1:5) {
                pa <- sample.gmm.params(k);
                obs <- simul.gmm.k( k * 1000, pa$pis, pa$mus, pa$sds )

                r <- evaluate.gs.em(pa,obs$x,d);
                df <- rbind(df, data.frame("dim.model"=c(k,k,k),
                                           "dim.est"=c(d,d,d),
                                           "method"=c("EM","GS.Mean","GS.MLE"),
                                           "pdf.diff"=r$pdf.diff,
                                           "llk.diff"=r$llk.diff,
                                           "time.sec"=r$time))

            }
        }
        print(df)
        pdf(paste("gmm",k,"pdf",sep=".",collapse=""),width=12,height=5)
        print( ggplot(df, aes(x=pdf.diff,y=llk.diff,colour=method)) +
               geom_point(alpha=0.4) + facet_grid(. ~ dim.est) );
        dev.off();
    }
}
```

Running `run.gibbs.eval()` should take a while, and it will eventually generate three pdf files.

(c) Interpret the results from part (b). What is the impact of d for differenr k? What are the differences between the three methods?