ANNE SABOURIN
8/11/2017

LAB SESSION 2: MINI-PROJECT ON GAUSSIAN MIXTURES

This mini-project is marked: you are supposed to hand out your work as a R notebook on the course's moodle before November, 26th. Details on the required format can be found there.

The aim of this lab session is to implement and compare the main computational methods surveyed in this course: EM, Variational Bayes and MCMC. We consider the Gaussian mixture example: given a dataset, the goal is to estimate the mixture distribution.To save time, A skeleton of the functions that you need to write to complete this project is given in the file `GM_functions.R`. Also all code chunks in the present assignment are gathered in the file `code_chunks2.R`

In the following, some extra `R` packages are needed:

```
##install.packages("MASS")
##install.packages("abind")
##install.packages("mnormt")
##install.packages("LaplacesDemon")
##install.packages("coda")
library(MASS)
library(abind)
library(mnormt)
library(LaplacesDemon)
library(coda)
```

1 Preliminaries

Recall the Gaussian mixture model on $\mathbb{R}^d$: The data $X = (X_1, \dots, X_n)$ is assumed to be *i.i.d.* according to the density

$$f(x|\rho_{1:k}, \mu_{1:k}, \Sigma_{1:k}) = \sum_{j=1}^{k} \rho_j \mathcal{N}(x|\mu_j, \Sigma_j) \ , \tag{1}$$

where k is the number of mixture components, $\mathcal{N}(x|\mu, \Sigma)$ denotes the Gaussian density with parameters μ, Σ, and $\rho_j \geq 0$ is the weight of component j, with $\sum_{j=1}^{k} p_j = 1$. The parameters $\mu_j \in \mathbb{R}^d$, Σ_j are respectively the center and the covariance matrix of component j.

This model can be rewritten using hidden variables $(\xi_i)_{i=1:n}$, with $\xi_i \in \{1, \dots, k\}$

$$\begin{aligned} &\xi_i \sim \text{Multinomial}(\rho) \\ &\mathcal{L}(X_i|\xi_i = j) = \mathcal{N}(\mu_j, \Sigma_j), \qquad j \in \{1, \dots, k\}. \end{aligned}$$

The parameter for the Gaussian mixture is thus $\theta = (\rho_{1:k}, \mu_{1:k}, \Sigma_{1:k})$, which we shall represent in `R` as three objects: `p, Mu, Sigma,` with `p` a vector of size k, `Mu` a $k \times d$ matrix and Σ a $d \times d \times k$ array.

We consider a synthetic dataset that we generate ourselves using the following code

```
d <- 2 ## dimension
K <- 3 ## number of mixture components
N <- 500 ## sample size
p <- c(3/10, 2/10, 5/10) ## weight parameter
Mu <- rbind(c(-1, -1), c(0, 1), c(1, -1)) ## centers
set.seed(1)
```

```
NN <- rmultinom(n = 1, size = N, prob = p)
## NN: number of points in each cluster
Sigma <- array(dim = c(d, d, K)) ## the covariance matrices
X <- matrix(ncol = d, nrow = 0) ## the dataset
for(j in 1:K){
   Sigma[, , j] <- rwishart(nu = 5, S = 0.05*diag(d))
}
for(j in 1:K){
   X <- rbind(X, mvrnorm(n=NN[j], mu = Mu[j, ], Sigma=Sigma[, , j]))
}

labs <- rep(0, N) ##vector of labels
count=1
for(j in 1:K){
   labs[count:(count+NN[j]-1)] <- j
   count <- count + NN[j]
}

#' Plot the labelled data
plot(X[,1], X[,2], col=labs)
```

In the sequel, the functions `gmllk, gmcdf, initem, draw_sd` will be repeatedly used. They are available in the file `GM_functions.R` with some comments.

2 EM Algorithm

We start with implementing the EM algorithm for Gaussian mixtures. Remind that the output of the algorithm is an approximation of the maximum likelihood estimates $(\widehat{\rho}, \widehat{\mu}_{1:k}, \widehat{\Sigma}_{1:k})$ respectively for the mixture weights, the mean parameters and the covariance parameters.

1. What are the coupled equations that are satisfied by the maximum likelihood estimates ?
2. Complete the code of functions `estep, mstep, emalgo` from file `GM_functions`.
3. Test your EM algorithm as follows:

```
Kfit <- 4 ## try with Kfit= 2,3,4,5 ...
outputem <- emalgo(x=X,k=Kfit, tol=1e-6)
## inspect the objective function (stopping criterion)
length(outputem$objective)

## Plot the (labelled) data
plot(X[,1], X[,2], col = labs, pch = 19)
## Add the starting points (from kmeans) to the plot
Init <- initem(X,Kfit)
points(Init$Mu[,1],Init$Mu[,2], col="orange",
      pch=18,cex=10*Init$p)
## add the centers from EM
points(outputem$last$Mu[,1],outputem$last$Mu[,2], col="blue",
      pch=18,cex=10*outputem$last$p)
## add the true centers
points(Mu[,1],Mu[,2], col="black",pch=8,
     cex=10*p)
for(j in 1:Kfit){
  ellips <- draw_sd(outputem$last$Mu[j,], outputem$last$Sigma[,,j])
    lines(ellips[1,], ellips[2,], col='blue')
  }

for(j in 1:K){
    ellips <- draw_sd(Mu[j,], Sigma[,,j])
    lines(ellips[1,], ellips[2,], col='black')
}
```

4. Check visually that the objective function to be maximized (the likelihood) increases at each iteration : plot the content of `outputem$objective`.

3 Variational Bayes

We turn to the Variational Bayes approach. As in the lectures, we use a mean field variational distribution $q(\rho, \mu_{1:k}, \Lambda_{1:k}) = q(\rho) \prod_{j=1}^{k} q(\mu_j, \Lambda_j)$, with $\Lambda_j = \Sigma_j^{-1}$. We choose a prior of a similar form:

- a Dirichlet distribution on ρ with parameter $\alpha = (\alpha_0, \dots, \alpha_0)$ for some $\alpha_0 > 0$.
- a Gaussian-Wishart distributions on each pair (ν_j, Λ_j) where $\Lambda_j = \Sigma_j^{-1}$, with parameter (ν_0, β_0, W_0).
- ρ and the pairs (μ_j, Λ_j) are independent

We have shown that the variational posterior distribution is of the same form as the prior, with different parameters $\alpha^* = (\alpha_1^*, \dots, \alpha_k^*)$, $\nu^* = (\nu_1^*, \dots, \nu_k^*)$, $\beta^* = (\beta_1^*, \dots, \beta_k^*)$, $W^* = (W_1^*, \dots, W_k^*)$.

1. What are the coupled equations satisfied by $\alpha^*, \nu^*, \beta^*, W^*$?
2. Complete the code of functions `vbMStep`, `vbEstep` and `vbalgo`. As a stopping criterion, the lower bound on the model likelihood is used,

$$\mathcal{F}(q) = \int_{z \in \mathcal{Z}} q(z) \log \frac{p(x, z)}{q(z)} \, \mathrm{d}z$$

   Where $z = (\xi_{1:n}, \rho_{1:k}, \mu_{1:k}, \Sigma_{1:k})$. In the present case, this lower bound has an explicit expression (see Bishop (2006), section 10.3.3). It is implemented in function `lowerbound`.

3. Test your algorithm with the following code

```
#' Bayesian model:
#' p ~ dirichlet(alpha); alpha = (alpha0, ... , alpha0)
#' [ xi | p ] ~ Multinomial(p)
#' [ mu_j | Lambda_j ] ~ Normal(m0, beta0 Lambda_j^(-1))
#' Lambda_j ~ Wishart(W0, nu0)
#' [ X| xi=j, mu, Lambda ] ~ Normal (mu_j, Lambda_j^(-1))

#' hyper-parameters : to be varied
alpha0 <- 0.1
m0 <- rep(0,2)
beta0 <- 0.1
W0 <- 1*diag(2)
```

```
nu0 <- 10
#' Run VB
#'
seed <- 10
set.seed(seed)
outputvb <- vbalgo(x=X,k=Kfit, alpha0 = alpha0, W0inv = solve(W0),
                   nu0 = nu0, m0 = m0, beta0=beta0, tol=1e-6)

#' plot the lowerbound over iterations
plot(outputvb$lowerbound)

#' show a summary of VB's output
T <- ncol(outputvb$alphamat)
outputvb$alphamat[,T]
outputvb$Marray[,,T]
```

4. As a first approximation we consider the posterior expectancy of the parameters in the variational approximation. Give the explicit expressions of

$$\widehat{\rho}_{vb} = \mathbb{E}_{q^*}(\rho), \quad \widehat{\mu}_{j,vb} = \mathbb{E}_{q^*}(\mu_j), \quad \widehat{\Sigma}_{j,vb} = (\mathbb{E}_{q^*}(\Lambda_j))^{-1},$$

as functions of the posterior predictive parameters. You may refer to Bishop (2006), Appendix B for expressions of expectancy's of classical distributions.

Plot a summary of the corresponding Gaussian mixture by completing the following code

```
#' Visual summary of VB's output :
#' posterior expectancy of each parameter
p_vb <- ## complete the code
    ## (variational posterior expectancy of mixture weights)
Mu_vb <- ## complete the code
    ## (variational posterior expectancy of mixture centers)
Sigma_vb <- array(dim=c(d,d,Kfit))
for(j in 1:Kfit){
    Sigma_vb[,,j] <- ## complete the code
    ## (variational posterior expectancy of mixture covariances)
}

## show the data, true centers and initial positions from K-means
graphics.off()
plot(X[,1], X[,2], col=labs)
points(Mu[,1],Mu[,2], col="black",pch=8,cex=10*p)
set.seed(seed)
Init <- initem(X,Kfit)
points(Init$Mu[,1],Init$Mu[,2], col="orange",pch=18,cex = 10*Init$p)
## Add a summary of the VB solution
nonneg <- which(p_vb>0.001)
for(j in nonneg){
    points(Mu_vb[j,1], Mu_vb[j,2], col="blue",
           pch=18,cex= 10 * p_vb[j])
    ellips <- draw_sd(mu = Mu_vb[j,],
                      sigma = Sigma_vb[,,j])
    lines(ellips[1,], ellips[2,], col='blue')
}
```

5. Study the influence of the hyper-parameter α_0: check that values less than one lead to 'sparse' solutions, in the sense that some mixture components are automatically granted negligible weights, so that the true number of components is automatically recovered (contrary to EM).

6. Study the influence of the other hyper-parameters.

4 Metropolis-Hastings algorithm

We now implement a Metropolis-Hastings algorithm for sampling the posterior distribution in the model described in Section 3.

Prior: The prior density is implemented in function `dprior` in the file `GM_functions.R`. It takes as argument a list of hyper-parameters `hpar`, see the function definition for details.

Proposal kernel: Denoting by $\theta^t = (\rho^t_{1:k}, \mu^t_{1:k}, \Sigma^t_{1:k})$ the current value of the parameter, we consider a proposal kernel $Q_{\text{MH}}(\theta^t, \theta^*)$, generating a proposal $\theta^* = (\rho^*, \mu^*_{1:k}, \Sigma^*_{1:k})$ as follows:

- $\rho^* \sim \mathcal{Diri}(\alpha_p \times \alpha^t)$ where $\alpha_p > 0$ is a concentration parameter fixed by the user.
- $\mu^*_j \sim \mathcal{N}(\mu^t_j, \sigma^2_\mu I_d)$ where σ^2_p is a variance parameter fixed by the user.
- $\Sigma^*_j \sim \text{Wishart}(\nu = \nu_\Sigma, W = (\nu_\Sigma)^{-1}\Sigma^t_j)$, where ν_Σ is a degree of freedom parameter fixed by user. Thus $\mathbb{E}_{Q_{\text{MH}}}(\Sigma^*_j|\Sigma^t_j) = \Sigma^t_j$ and the distribution of Σ^*_j is more peaked around Σ^t_j for large values of ν_Σ.

1. Complete the code for the function `rproposal` in file `GM_functions.R` to generate such a proposal.

2. Complete the code for the Metropolis-Hastings sampler `MHsample`.

3. Test your code on the following example (fix `nsample` to a lower value for debugging). For comparison purposes, the hyper-parameter values should be the same as those used with Variational Bayes.

```
Kmc <- Kfit ## try with different values
init <- initem(x=X, k=Kmc)

hpar <- list( alpha0=rep(alpha0, Kmc),
        m0 = rep(0, d), beta0 = beta0,
        W0 = W0, nu0 = nu0)

ppar <- list(var_Mu = 0.001,
         nu_Sigma = 500,
         alpha_p = 500)


set.seed(1)
pct <- proc.time()
outputmh <- MHsample(x=X, k=Kmc, nsample= 3000,
                    init=init, hpar=hpar, ppar=ppar)
newpct <- proc.time()
elapsed <- newpct - pct
elapsed
outputmh$naccept ## should not be ridiculously low.
```

For convergence diagnostics, tracing the evolution of a single parameter (such as $p[1]$) is not relevant because of possible re-labelling of the mixture components (the model is not identifiable). However relevant numerical summaries can be constructed, such as the value of the cumulative distribution function (cdf) at a given point $x = (x_1, \dots, x_d)$,

$$F(y|\rho^t, \mu^t_{1:k}, \Sigma^t_{1:k}) = \mathbb{P}(X_1 \le y_1, \dots, X_d \le y_d \,|\, \rho^t, \mu^t_{1:k}, \Sigma^t_{1:k}), \quad t \in \{1, \dots, N_{sample}\},$$

where N_{sample} is the number of iterations

4. To obtain such a time series, complete the code of function `cdfTrace` from file `GM_functions.R`. Notice that parameters `thin` and `burnin` allow respectively to keep only one out of '`thin`' sample and to discard the first `burnin` samples.

5. The Heidelberger and Welch's test is a convergence test based on the stationarity hypothesis (see the help for function `heidel.`**`diag`** from package `coda`). Use the function `heidel.`**`diag`** from package `coda` in combination with `cdfTrace` to propose a reasonable number of iterations to achieve convergence. Use various values of x as argument of `cdfTrace`.
N.B: the functions takes as argument a `mcmc` object. Any vector `y` may be coerced into an mcmc object via: `y <- mcmc(y).`

6. Generate three different chains `outputmh1,outputmh2, outputmh3` with different starting values and use the Gelman and Rubin's diagnostic (`coda` functions `gelman.`**`diag`** and `gelman.`**`plot`**) as an additional convergence check. Discuss your results in combination with the previous ones to determine a reasonable number of iterations.

7. Remind that the predictive density is the posterior mean

$$f_{\text{pred}}(y) = \int f(y|\rho, \mu_{1:k}, \Sigma_{1:k})\pi(\rho, \mu_{1:k}, \Sigma_{1:k}|x_{1:n})\,\mathrm{d}\rho\,\mathrm{d}\mu_{1:k}\,\mathrm{d}\Sigma_{1:k},$$

which can be estimated from the MH sample by computing the empirical mean of the density over the MH output,

$$\widehat{f}_{\text{MH}}(y) = \frac{1}{M}\sum_{t=1}^{M} f(y|\rho^t, \mu^t_{1:k}, \Sigma^t_{1:k}),$$

where M is the number of remaining samples after discarding the burn-in period and thinning. Notice that thinning does not make the estimator better, it is just a convenient way to reduce the computational time when the number of samples is large and when consecutive samples are highly correlated.

The goal of this question is to plot the numerical approximation $\widehat{f}_{\text{MH}}(y)$ on a grid of size 20×20, together with the true density.

- Complete the code of function `MHpredictive`, which returns $\widehat{f}_{\text{MH}}(x)$.
- Complete the following code chunk in order to plot the desired result together with the true density. Define `outputmh` as one of the three previously generated chains with good p-values for the Heidelberger and Welch's test. Notice that the `wrapper` function from file `GM_functions.R` is a convenient auxiliary allowing to apply a function on a grid via the `R` function `outer`.

```
xx <- seq(-2,2,length.out=20)
yy <- xx
dtrue <- outer(X= xx, Y=yy,
          FUN = function(x,y){
             wrapper(x=x, y=y,
                   FUN=function(u,v){
                     exp(gmllk(x = c(u,v), Mu = Mu,
                     Sigma = Sigma, p = p))
                   })
          })

dpredmh <- outer(X= xx, Y=yy,
          FUN = function(x,y){
             wrapper(x = x, y = y,
                   FUN =function(u,v){
                 ## complete the code })
          })

breaks <- c(seq(0.01,0.09, length.out=5),seq(0.1,0.3,length.out=5))
nbreaks <- length(breaks)
contour(xx,yy, z = dtrue, nlevels=nbreaks, levels = breaks)
contour(xx,yy, z = dpredmh, nlevels=nbreaks, levels = breaks,
        add=TRUE, col='red')
```

Comment your results.

5 Predictive distributions versus maximum likelihood distribution

In this section we focus on the probability of an excess of a threshold $u \in \mathbb{R}$:

$$\varphi(u, \rho, \mu_{1:k}, \Sigma_{1:k}) = 1 - F(u|\rho, \mu_{1:k}, \Sigma_{1:k}) = \mathbb{P}(X_1 > u \text{ or } \ldots \text{ or } X_d > u|\rho, \mu_{1:k}, \Sigma_{1:k}).$$

where $F(y|\rho, \mu_{1:k}, \Sigma_{1:k})$ is the cdf for the Gaussian Mixture. The latter is already coded as function `gmcdf` in file `GM_functions.R`. Our goal is to compare the performance of the estimators obtained by EM (maximum likelihood), Variational Bayes and Metropolis-Hastings, namely

$$\widehat{\varphi}_1(u) = \varphi(u, \widehat{\rho}, \widehat{\mu}_{1:k}, \widehat{\Sigma}_{1:k}),$$
$$\widehat{\varphi}_2(u) = \mathbb{E}\Big[\varphi(u, \rho, \mu_{1:k}, \Sigma_{1:k})\Big] \text{ with } (\rho, \mu_{1:k}, \Sigma_{1:k}) \sim Q^*$$
$$\widehat{\varphi}_3(u) = \frac{1}{M}\sum_{t=1}^{M} \varphi(u, \rho^t, \mu^t_{1:k}, \Sigma^t_{1:k})$$

where $\widehat{\rho}, \widehat{\mu}_{1:k}, \widehat{\Sigma}_{1:k}$ are the maximum likelihood estimates issued by the EM algorithm, Q^* is the posterior variational distribution and where M is the number of remaining samples after discarding the burn-in period and thinning.

The estimators $\widehat{\varphi}_2, \widehat{\varphi}_3$ are numerical approximations of $1 - F_{pred}(u)$ where F_{pred} is the predictive cdf,

$$F_{pred}(x) = \int F(y|\rho, \mu_{1:k}, \Sigma_{1:k})\pi(\rho, \mu_{1:k}, \Sigma_{1:k}|x_{1:n})\, \mathrm{d}\rho\, \mathrm{d}\mu_{1:k}\, \mathrm{d}\Sigma_{1:k}$$

To summarize:

- The true φ is easily computed via function `gmcdf`
- $\widehat{\varphi}_1$ is computed in a similar way using the the output of EM as argument of `gmcdf`
- It can be shown (See Bishop (2006), Section 10.2.3) that the variational predictive distribution is a mixture of multivariate Student distributions with parameters that are functions of the optimized parameters $(\alpha^*, m_j^*, \beta_j^*, W_j^*, \nu_j^*), j \in \{1, \ldots, k\}$ of the variational posterior distribution. The cdf of this mixture is implemented in function `vbPredictiveCdf` from file `GM_functions.R`. This function is the main ingredient for computing $\widehat{\varphi}_2$.
- $\widehat{\varphi}_3$ can be computed similarly to the posterior predictive density (Section 4.7) via the function `MHpredictiveCdf` (to be completed)

1. Complete the code of function `MHpredictiveCdf`
2. We consider a range of thresholds on the diagonal line, $u = (x, x)$, for $x \in [-3, 3]$. Complete the following code chunk in order to plot on the same graph, as a function of x,

$$\varphi((x,x)|\rho, \mu_{1:k}, \Sigma_{1:k}), \quad \widehat{\varphi}_1(x,x), \quad \widehat{\varphi}_2(x,x), \quad \widehat{\varphi}_3(x,x).$$

Comment your results.

```
Pexcess <- rep(0,10)
Pexcess_em <- Pexcess; Pexcess_vb <- Pexcess; Pexcess_mh <- Pexcess
thres_vect <- seq(-3, 3, length.out=30)
for(i in seq_along(thres_vect)){
threshold <- rep(thres_vect[i], 2)
Pexcess[i] <- 1 - gmcdf(x = threshold, Mu = Mu, Sigma=Sigma, p=p)
Pexcess_em[i] <- ## complete the code:
             ##maximum likelihood estimator using EM output
Pexcess_vb[i] <- ## complete the code:
   ## posterior predictive estimator using VB output:
   ## use vbPredictiveCdf

Pexcess_mh[i] <- ## complete the code:
   ## posterior predictive estimator using MH output:
   ## use MHpredictiveCdf.

ylim <- range(Pexcess, Pexcess_em,Pexcess_vb)
plot(thres_vect,Pexcess, ylim = ylim)
lines(thres_vect, Pexcess_vb, col='red')
lines(thres_vect, Pexcess_em, col='blue')
lines(thres_vect, Pexcess_mh, col='green')
```

3. Consider now the tails of the mixture distribution: replace the third line in the above code chunk with

```
   thres_vect <- seq(1, 5, length.out=30)
```

Comment your results, in particular explain the behavior of the Variational Bayes estimator.

4. We now focus on $\widehat{\varphi}_3$. Plot on the same graph $\varphi((x,x)|\rho, \mu_{1:k}, \Sigma_{1:k})$ and $\widehat{\varphi}_1((x,x)$ together with posterior 90% credible sets obtained with the empirical quantiles of the time series $\varphi((x,x)|\rho^t, \mu^t_{1:k}, \Sigma^t_{1:k})$, $t \in 1, \ldots, M$ where $(\rho^t, \mu^t_{1:k}, \Sigma^t_{1:k}), t \leq M$ is the output of the MH algorithm after thinning and discarding an appropriate burn-in period. Comment the results.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer. 3, 4, 8