MINI-PROJECT: 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. The required format is a single .zip archive, named after you (*e.g.* DupontJean.zip) containing

- The notebook source code: a .Rmd or .R file if you worked with Rstudio or a .ipynb file if you worked with Jupter with the R kernel.
- The compiled version of the notebook: a html or pdf file (html is preferred but may cause size limit issues). This version should contain all figures and results generated by the code.

The aim of this lab session is to implement and compare the main computational methods surveyed in this course: 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 `fonctions_GM.R`. Also all code chunks in the present assignment are gathered in the file `code_chunks2019.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 $\sum_{j=1}^{k} \rho_j = 1$. 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 $\xi_i \in \{1, \dots, k\}$.

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

It is mathematically convenient to represent the multimodal variable ξ_i by a binary vector z_i of size k. 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 (standing for ρ), `Mu` a $k \times d$ matrix and Σ a $d \times d \times k$ array.

For the sake of simplicity, we consider in this project a synthetic dataset that we generate ourselves. This allows you in particular to assess the correctness of your work, by comparison with the ground truth.

1. Generate a bi-variate dataset $X_{1:N}$ of size $N = 500$ from a Gaussian mixture with 3 components, with parameters $(\mu_{1:3} = (0,0),(1,0),(0,1))$, with covariance matrices $\Sigma_{1:3}$ randomly drawn independently according to a Wishart distribution with parameter $(\nu = 4, S = \mathbf{I}_2)$. To do so, use the R function `rwishart` and set the seed of the random generator to 1 (by writing `set.seed(1)` immediatly before generating the three covariance matrices). In order to keep track of the assignment of each X_i, store the hidden variables ξ_i's used to generate the X_i's.

2. Plot the generated data using one color for each mixture component, *i.e.* any point X_i such that $\xi_i = j$ $(j = 1,\dots,3)$ should be assigned the j^{th} color from R's color palette (NB: this is achieved with the option `col = j` inside the plot function.

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

2 Variational Bayes

We start with the Variational Bayes approach. As in the lectures, we use a mean field variational distribution $q(z,\rho,\mu_{1:k},\Lambda_{1:k}) = q(z)q(\rho,\mu,\Lambda)$ and we choose a prior as follows:

- 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 $\alpha^* = (\alpha_1^*,\dots,\alpha_k^*)$, $\nu^* = (\nu_1^*,\dots,\nu_k^*)$, $\beta^* = (\beta_1^*,\dots,\beta_k^*)$, $W^* = (W_1^*,\dots,W_k^*)$. The coupled equations satisfied by $\alpha^*,\nu^*,\beta^*,W^*$ are given in Bishop (2006), available on line here: https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognitio and-Machine-Learning-2006.pdf, pp. 478-479, equations (10.59) to (10.68)

1. Complete the code of functions `vbMStep`, `vbEstep` and `vbalgo`. To choose the starting values automatically, use the function `initPar` from the file `fonctions_GM`, which runs a simple k-means. As a stopping criterion, use *e.g.* the squared L_2 norm between two successive values of the parameters (when viewed as a concatenated single vector).

2. 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
#'
```

```r
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 Stopping criteria over iterations
plot(outputvb$stopCriteria)

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

3. To assess the quality of the ouput, we first compare the ground truth (true mixture density), with the density of the mixture associated with point estimates $(\widehat{\rho}, \widehat{\mu}, (\widehat{\Lambda})^{-1})$ constructed from the VB output. Namely , we choose as a parameter estimate, the posterior expectancy of this parameter in the variational approximation.

   Write down the expression of $(\widehat{\rho}, \widehat{\mu}, \widehat{\Lambda})$ in this framework, as a function of the VB output. 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

```r
#' Visual summary of VB's output :
#' posterior expectancy of each parameter
p_vb <- ## complete the code

Mu_vb <- ## complete the code

Sigma_vb <- array(dim=c(d,d,Kfit))
for(j in 1:Kfit){

   Sigma_vb[,,j] <- ## complete the code

}

## 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 <- initPar(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')
}
```

4. 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).

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

3 Metropolis-Hastings algorithm

We now implement a Metropolis-Hastings algorithm for sampling the posterior distribution in the model described in Section 2.

Prior: The prior density is implemented in function `dprior` in the file `fonctions_GM.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:

- $\rho^* \sim \mathcal{D}iri(\alpha_p \times \alpha^t)$ 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 `fonctions_GM.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 <- initPar(x=X, k=Kmc)

hpar <- list( alpha0= alpha0,
        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 `fonctions_GM.R`. Notice that parameters `thin` and `burnin` allow respectively to keep only one out of '`thin`' samples 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 `fonctions_GM.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.

4 Predictive distributions versus maximum likelihood distribution

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

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

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 `fonctions_GM.R`. Our goal is to compare the performance of the estimators obtained by Variational Bayes and Metropolis-Hastings, namely

$$\widehat{\varphi}_2(u) = \mathbb{E}\Big[\varphi(u, \rho, \mu_{1:k}, \Sigma_{1:k})\Big] \text{ with } (\rho, \mu_{1:k}, \Sigma_{1:k}) \sim Q^*$$

$$\widehat{\varphi}_3(u) = \frac{1}{M}\sum_{t=1}^{M} \varphi(u, \rho^t, \mu^t_{1:k}, \Sigma^t_{1:k})$$

where 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`

- 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 $(\alpha^*, m_j^*, \beta_j^*, W_j^*, \nu_j^*), j \in \{1, \dots, k\}$ of the variational posterior distribution. The cdf of this mixture is implemented in function `vbPredictiveCdf` from file `fonctions_GM.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 3.7) 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 $x \in [-1, 4]$. Complete the following code chunk in order to plot on the same graph, as a function of x,

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

Comment your results.

```
Pexcess <- rep(0,10)
Pexcess_vb <- Pexcess; Pexcess_mh <- Pexcess
thres_vect <- seq(-1, 4, 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_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_vb, Pexcess_mh)
plot(thres_vect,Pexcess, ylim = ylim)
lines(thres_vect, Pexcess_vb, col='red')
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 $\widehat{\varphi}_3((x,x)$ together with posterior 90% credible sets obtained with the empirical quantiles of the time series $\varphi((x,x)|\rho^t, \mu_{1:k}^t, \Sigma_{1:k}^t)$, $t \in 1, \dots, M$ where $(\rho^t, \mu_{1:k}^t, \Sigma_{1:k}^t), t \leq M$ 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. 2, 3, 7