TDDE07 Bayesian Learning - Lab 3

Erik Linder-Norén - erino397

2017-05-09

## 1. Normal model, mixture of normal model with semi-conjugate prior.

### (a) Normal model

Plotting the precipitation data and the density of that data resulted in Figure 1.

![Precipitation and density](data:application/pdf;base64,) $\pagebreak$

The convergence of the sampled parameters and can be seen in Figure 2, where the mean of sequential draws during Gibbs sampling are calculated and visualized to show the autocorrelation between those draws, and how the sampled parameters approaches the true respective values.

![Predictions beta priors](data:application/pdf;base64,)

Predictions beta priors

$\pagebreak$

### (b) Mixture normal model.

The convergence of the parameters and can be seen in Figure 3.

![Mixture normal model](data:application/pdf;base64,)

Mixture normal model

$\pagebreak$

### (c) Graphical comparison.

A comparison between the kernel density estimate of the data, a normal approximation by Gibbs sampling with parameters derived in (a) and a mixture of normals from (b) can be seen in Figure 4. Comparing a normal approximation of the data density with a mixture of normals approximation of the data - it is clear that a mixture of normals is a much better approximation of the true density of the data.

![Mixture normal model](data:application/pdf;base64,)

Mixture normal model

$\pagebreak$ ## 2. Probit regression ### (c) ![Comparison between predictive distribution of a normal approximation and the predictive distribution of probit regression gained by Gibbs sampling](data:application/pdf;base64,)

$\pagebreak$

## Assignment 1

require(MASS)  
require(geoR)  
  
grid\_w = 5  
grid\_h = 4  
  
# Lab 3 - Assignment 1  
  
data = read.table("data/rainfall.txt", header=FALSE)[,1]  
  
n = length(data)  
  
# (a)  
  
pdf("plots/3\_1\_1\_precipitation.pdf")  
  
par(mfrow=c(2,1))  
  
# Plot the precipitation  
plot(data,   
 type="h",   
 lwd=2,  
 ylab="Precipitation",  
 xlab="Time",  
 xaxt="n",  
 col="black",  
 main="Precipitation 1948-1983")  
  
axis(1,   
 at=seq(0, n, n/(1983-1948)),   
 labels=seq(1948, 1983))  
  
# Plot the precipitation density  
prec\_density = density(data)  
plot(prec\_density,   
 type="l",   
 lwd=2,  
 xlab="Precipitation",  
 ylab="Density",  
 main="The Daily Precipitation")  
  
dev.off()  
  
data\_mean = mean(data)  
  
# Prior parameters for sigma (variance)  
v0 = 1  
sigma\_sq0 = 1 / v0  
  
n\_draws = 4000  
  
# Initial value for sigma  
sigma\_sq = rinvchisq(n=1, v0, sigma\_sq0)  
  
  
gibbs\_draws = matrix(0,n\_draws,2)  
for(i in 1:n\_draws){  
 mu = rnorm(n=1, mean=data\_mean, sd=sqrt(sigma\_sq/n))  
 sigma\_sq = rinvchisq(n=1, v0 + n, (v0\*sigma\_sq0 + sum((data - mu)^2))/(n + v0))   
 gibbs\_draws[i,] = c(mu, sigma\_sq)  
}  
  
  
mean\_draws = gibbs\_draws[,1]  
var\_draws = gibbs\_draws[,2]  
  
  
# Calculate mean of batches of 2 draws to visualize the  
# auto correlation between sequential draws  
mean\_means = c()  
mean\_vars = c()  
for (i in 1:n\_draws){  
 if(i%%2 == 0){  
 mean\_means = c(mean\_means, mean(mean\_draws[i-1:i]))  
 mean\_vars = c(mean\_vars, mean(var\_draws[i-1:i]))  
 }  
}  
  
# Plots displaying convergence of the Normal hyper   
# parameters during sampling  
  
pdf("plots/3\_1\_1\_gibbs\_sampl\_conv.pdf")  
  
par(mfrow=c(2,1))  
  
# Plot the auto correlation (convergence) between draws of mu  
min\_mean = min(mean\_means)  
max\_mean = max(mean\_means)  
plot(mean\_means,   
 type="l",   
 ylim=c(min\_mean, max\_mean),   
 cex=.1,  
 lwd=2,  
 main=expression(paste("Convergence of Gibbs Sampling ", "(", mu, ")", sep=" ")),  
 xlab="Batches of sequential draws",  
 ylab=expression(paste("Mean of seq. draws of ", mu, sep=" ")))  
  
  
  
# Plot the auto correlation (convergence) between draws of sigma  
min\_var = min(mean\_vars)  
max\_var = max(mean\_vars)  
plot(mean\_vars,   
 type="l",   
 ylim=c(min\_var, max\_var),   
 cex=.1,  
 lwd=2,  
 main=expression(paste("Convergence of Gibbs Sampling ", "(", sigma^2, ")", sep=" ")),  
 xlab="Batches of sequential draws",  
 ylab=expression(paste("Mean of seq. draws of ", sigma^2, sep=" ")))  
  
dev.off()  
  
# (c)  
  
# Kernel density estimate  
pdf("plots/3\_1\_3\_dens\_comp.pdf")  
  
par(mfrow=c(3,1))  
  
kernel\_density = density(data)  
  
plot(kernel\_density$x,   
 kernel\_density$y,   
 type="l",  
 cex=.1,  
 lwd=2,  
 ylab="Density",  
 xlab="Precipitation",  
 main="Rainfall: Kernel density estimate")  
  
# Normal density from (a)  
  
mean = mean(mean\_draws)  
std\_dev = sqrt(mean(var\_draws)/n)  
  
x\_grid = seq(mean - 2, mean + 2, 0.0001)  
  
normal\_density = dnorm(x\_grid, mean=mean, sd=std\_dev)  
  
plot(x\_grid,   
 normal\_density,   
 type="l",  
 cex=.1,  
 lwd=2,  
 ylab="Density",  
 xlab="Precipitation",  
 main="Rainfall: Normal density")  
  
# Mixture of normals from (b)  
# (run './template/gaussian\_mixture.R' first)  
  
hist(x, breaks = 20, cex=.1, border="lightgray", freq = FALSE, xlim = c(xGridMin,xGridMax), xlab="Precipitation", ylab="Density", main = "Rainfall: Mixture of Normals")  
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "black")  
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "gray")  
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=c("lightgray","black","gray"), lwd = 2)  
  
  
dev.off()

$\pagebreak$

## Assignment 2

require(mvtnorm)  
require(msm)  
library(LaplacesDemon)  
  
grid\_w = 6  
grid\_h = 5  
  
# -------  
# Lab 3  
# -------  
  
  
# Read data  
data <- read.table("./data/WomenWork.dat", header = TRUE)  
  
y <- as.vector(data$Work)  
X <- as.matrix(data[, 2:ncol(data)])  
  
# Data spec.  
n\_features = ncol(X)  
n\_samples = nrow(X)  
  
tau = 10  
  
  
# -------------  
# (a) and (b)  
# -------------  
  
  
# Beta prior parameters  
mu0 = rep(0, n\_features)  
covar0 = diag(tau^2, n\_features)  
  
draw\_beta = function(y) {  
   
 X\_X = t(X) %\*% X  
   
 # Least squares approximate of beta  
 beta\_hat = ginv(X\_X) %\*% t(X) %\*% y  
   
 # Posterior parameters  
 mu\_n = ginv(X\_X + covar0)%\*%(X\_X%\*%beta\_hat+covar0%\*%mu0)  
 covar\_n = X\_X + covar0  
   
 # Assuming sigma\_sq = 1  
 b\_draw = rmvnorm(1, mean=mu\_n, sigma=ginv(covar\_n))  
   
 return(b\_draw)  
}  
  
draw\_u = function(beta) {  
   
 # Mean of predictive distr.  
 regr\_mean = X %\*% t(beta)  
   
 u = rep(0, n\_samples)  
 for(i in 1:n\_samples) {  
 y\_i = y[i]  
   
 if(y\_i == 0){  
 # Truncate [-inf, 0)  
 u[i] = rtnorm(n=1, mean=regr\_mean[i], sd=1, upper=0)  
 }else{  
 # Truncate (0, inf]  
 u[i] = rtnorm(n=1, mean=regr\_mean[i], sd=1, lower=0)  
 }  
 }  
   
 return(u)  
}  
  
# Initial prediction  
u = rnorm(n\_samples, covar0)  
  
n\_draws = 500  
beta\_draws = matrix(0, n\_draws, n\_features)  
u\_draws = matrix(0, n\_draws, n\_samples)  
for(i in 1:n\_draws) {  
 beta = draw\_beta(u)  
 u = draw\_u(beta)  
 beta\_draws[i,] = beta  
 u\_draws[i,] = u  
}  
  
# Avoid first 10% of the draws  
burn\_in = floor(n\_draws / 10)  
beta\_draws = beta\_draws[burn\_in:nrow(beta\_draws),]  
  
beta\_mean = colMeans(beta\_draws)  
beta\_cov = cov(beta\_draws)  
  
# -----  
# (c)  
# -----  
  
  
# Calculate the log posterior  
LogPosteriorProbit <- function(betas, y, X, mu, Sigma){  
   
 # Multiply data by parameters to get predictions  
 predictions <- X%\*%betas;  
   
 # Log likelihood (for probit)   
 log\_likelihood = sum(y\*pnorm(predictions, log.p = TRUE) + (1-y)\*pnorm(predictions, log.p = TRUE, lower.tail = FALSE))  
   
 # Log prior  
 log\_prior <- dmvnorm(betas, mu0, covar0, log=TRUE);  
   
 # Sum of log likelihood and log prior is log posterior  
 return(log\_likelihood + log\_prior)  
}  
  
log\_posterior = LogPosteriorProbit  
  
# Initialize as zeros  
init\_betas = rep(0, n\_features)  
  
opt\_results = optim(init\_betas,  
 log\_posterior,  
 gr=NULL,  
 y,  
 X,  
 mu0,  
 covar0,  
 method=c("BFGS"),  
 control=list(fnscale=-1),  
 hessian=TRUE)  
  
# Posterior mode (beta hat)  
post\_mode = opt\_results$par  
# Posterior covariance (J^-1(beta hat))  
post\_cov = -solve(opt\_results$hessian)  
  
# Sample to predict  
sample = c(constant=1,  
 husband=10,  
 edu\_years=8,  
 exp\_years1=10,  
 exp\_years2=(10/10^2),  
 age=40,  
 n\_small\_child=1,  
 n\_big\_child=1)  
  
get\_pred = function(beta){  
 e = exp(sample%\*%beta)  
 # Calculate the probability (bernoulli parameter)  
 p = e / (1 + e)  
 # Draw a y prediction  
 y\_draw = rbern(n=1, prob=p)  
}  
  
n\_draws = 100  
y\_draws\_2 = c() # As in lab 2  
y\_draws\_3 = c() # As in lab 3  
for (i in 1:n\_draws){  
 # Get prediction according to lab 2  
 beta1 = as.vector(rmvnorm(n=1, mean=post\_mode, sigma=post\_cov))  
 y\_draw = get\_pred(beta1)  
 y\_draws\_2 = c(y\_draws\_2, y\_draw)  
  
 # Get prediction according to lab 3  
 beta2 = as.vector(rmvnorm(n=1, mean=beta\_mean, sigma=beta\_cov))  
 y\_draw = get\_pred(beta1)  
 y\_draws\_3 = c(y\_draws\_3, y\_draw)  
}  
  
pdf("plots/3\_2\_3\_pred\_distr\_comp.pdf", width=grid\_w, height=grid\_h)  
  
prob\_density = density(y\_draws\_2)  
plot(prob\_density,  
 type="l",  
 lwd=2,  
 xlim=c(0,1),  
 ylim=c(0,2.5),  
 ylab="Density",  
 xlab="Prediction (0 = not working, 1 = working)",  
 main="Predictive distribution for sample",  
 col="black",  
 cex.main=.9,  
 cex.lab=.9,  
 cex.axis=.8)  
  
prob\_density = density(y\_draws\_3)  
lines(prob\_density, col="gray", lwd=2)  
  
legend("topright",   
 legend = c("Normal Approximation","Gibbs Probit"),  
 fill = c("black", "gray"),  
 inset = 0.02)  
  
dev.off()

$\pagebreak$

## Modifications to Mattias impl. of Mixture of Normals

# Estimating a simple mixture of normals  
# Author: Mattias Villani, IDA, Linköping University. http://mattiasvillani.com  
  
########## BEGIN USER INPUT #################  
# Data options  
data(faithful)  
rawData <- faithful  
x <- as.matrix(rawData['eruptions'])  
  
# Lab 3 -   
data = read.table("data/rainfall.txt", header=FALSE)[,1]  
x = as.matrix(data)  
  
# Model options  
nComp <- 2 # Number of mixture components  
  
# Prior options  
alpha <- 10\*rep(1,nComp) # Dirichlet(alpha)  
muPrior <- rep(0,nComp) # Prior mean of theta  
tau2Prior <- rep(10,nComp) # Prior std theta  
sigma2\_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)  
nu0 <- rep(4,nComp) # degrees of freedom for prior on sigma2  
  
# MCMC options  
nIter <- 100 # Number of Gibbs sampling draws  
  
# Plotting options  
plotFit <- TRUE  
lineColors <- c("blue", "green", "magenta", 'yellow')  
sleepTime <- 0.1 # Adding sleep time between iterations for plotting  
################ END USER INPUT ###############  
  
###### Defining a function that simulates from the   
rScaledInvChi2 <- function(n, df, scale){  
 return((df\*scale)/rchisq(n,df=df))  
}  
  
####### Defining a function that simulates from a Dirichlet distribution  
rDirichlet <- function(param){  
 nCat <- length(param)  
 thetaDraws <- matrix(NA,nCat,1)  
 for (j in 1:nCat){  
 thetaDraws[j] <- rgamma(1,param[j],1)  
 }  
 thetaDraws = thetaDraws/sum(thetaDraws) # Diving every column of ThetaDraws by the sum of the elements in that column.  
 return(thetaDraws)  
}  
  
# Simple function that converts between two different representations of the mixture allocation  
S2alloc <- function(S){  
 n <- dim(S)[1]  
 alloc <- rep(0,n)  
 for (i in 1:n){  
 alloc[i] <- which(S[i,] == 1)  
 }  
 return(alloc)  
}  
  
# Initial value for the MCMC  
nObs <- length(x)  
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-nComp matrix with component allocations.  
theta <- quantile(x, probs = seq(0,1,length = nComp))  
sigma2 <- rep(var(x),nComp)  
probObsInComp <- rep(NA, nComp)  
  
# Setting up the plot  
xGrid <- seq(min(x)-1\*apply(x,2,sd),max(x)+1\*apply(x,2,sd),length = 100)  
xGridMin <- min(xGrid)  
xGridMax <- max(xGrid)  
mixDensMean <- rep(0,length(xGrid))  
effIterCount <- 0  
ylim <- c(0,2\*max(hist(x)$density))  
  
gibbs\_thetas = matrix(0,nIter,2)  
gibbs\_sigmas = matrix(0,nIter,2)  
for (k in 1:nIter){  
 message(paste('Iteration number:',k))  
 alloc <- S2alloc(S) # Just a function that converts between different representations of the group allocations  
 nAlloc <- colSums(S)  
 print(nAlloc)  
 # Update components probabilities  
 w <- rDirichlet(alpha + nAlloc)  
   
 # Update theta's  
 for (j in 1:nComp){  
 precPrior <- 1/tau2Prior[j]  
 precData <- nAlloc[j]/sigma2[j]  
 precPost <- precPrior + precData  
 wPrior <- precPrior/precPost  
 muPost <- wPrior\*muPrior + (1-wPrior)\*mean(x[alloc == j])  
 tau2Post <- 1/precPost  
 theta[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))  
 }  
   
 gibbs\_thetas[k, ] = theta  
   
 # Update sigma2's  
 for (j in 1:nComp){  
 sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j], scale = (nu0[j]\*sigma2\_0[j] + sum((x[alloc == j] - theta[j])^2))/(nu0[j] + nAlloc[j]))  
 }  
   
 gibbs\_sigmas[k,] = sigma2  
   
 # Update allocation  
 for (i in 1:nObs){  
 for (j in 1:nComp){  
 probObsInComp[j] <- w[j]\*dnorm(x[i], mean = theta[j], sd = sqrt(sigma2[j]))  
 }  
 S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))  
 }  
   
 # Printing the fitted density against data histogram  
 if (plotFit && (k%%1 ==0)){  
 effIterCount <- effIterCount + 1  
 hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number",k), ylim = ylim)  
 mixDens <- rep(0,length(xGrid))  
 components <- c()  
 for (j in 1:nComp){  
 compDens <- dnorm(xGrid,theta[j],sd = sqrt(sigma2[j]))  
 mixDens <- mixDens + w[j]\*compDens  
 lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])  
 components[j] <- paste("Component ",j)  
 }  
 mixDensMean <- ((effIterCount-1)\*mixDensMean + mixDens)/effIterCount  
   
 lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')  
 legend("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),   
 col = c("black",lineColors[1:nComp], 'red'), lwd = 2)  
 Sys.sleep(sleepTime)  
 }  
   
}  
  
  
# Calculate mean of batches of 2 draws to visualize the  
# auto correlation between sequential draws  
t1 = c()  
t2 = c()  
s1 = c()  
s2 = c()  
for (i in 1:nIter){  
 if(i%%2 == 0){  
 t1 = c(t1, mean(gibbs\_thetas[,1][i-1:i]))  
 t2 = c(t2, mean(gibbs\_thetas[,2][i-1:i]))  
 s1 = c(s1, mean(gibbs\_sigmas[,1][i-1:i]))  
 s2 = c(s2, mean(gibbs\_sigmas[,2][i-1:i]))  
 }  
}  
  
  
# Plots displaying convergence of the Normal hyper   
# parameters during sampling  
  
pdf("plots/3\_1\_2\_mixt\_conv.pdf")  
  
par(mfrow=c(2,1))  
  
# Plot the auto correlation (convergence) between draws of mu  
min\_t = min(c(min(t1), min(t2)))  
max\_t = max(c(max(t1), max(t2)))  
plot(t1,   
 type="l",   
 ylim=c(min\_t, max\_t),   
 cex=.1,  
 lwd=2,  
 main=expression(paste("Convergence of Gibbs Sampling ", "(", theta, ")", sep=" ")),  
 xlab="Batches of sequential draws",  
 ylab=expression(paste("Mean of seq. draws of ", theta, sep=" ")))  
  
lines(t2, lwd=2, col="gray")  
  
legend("topright",   
 box.lty = 1,   
 legend = c(expression(paste(theta, " (1)", sep=" ")),  
 expression(paste(theta, " (2)", sep=" "))),   
 col=c("black","gray"),   
 lwd = 2)  
  
# Plot the auto correlation (convergence) between draws of sigma  
min\_s = min(c(min(s1), min(s2)))  
max\_s = max(c(max(s1), max(s2)))  
plot(s1,   
 type="l",   
 ylim=c(min\_s, max\_s),   
 cex=.1,  
 lwd=2,  
 main=expression(paste("Convergence of Gibbs Sampling ", "(", sigma^2, ")", sep=" ")),  
 xlab="Batches of sequential draws",  
 ylab=expression(paste("Mean of seq. draws of ", sigma^2, sep=" ")))  
  
lines(s2, lwd=2, col="gray")  
  
legend("topright",   
 box.lty = 1,   
 legend = c(expression(paste(sigma^2, " (1)", sep=" ")),  
 expression(paste(sigma^2, " (2)", sep=" "))),   
 col=c("black","gray"),   
 lwd = 2)  
  
dev.off()  
  
grid\_w = 6  
grid\_h = 5  
  
pdf("plots/3\_1\_3\_mixt\_norm.pdf", width=grid\_w, height=grid\_h)  
  
hist(x, breaks = 20, cex=.1, border="lightgray", freq = FALSE, xlim = c(xGridMin,xGridMax), xlab="Precipitation", ylab="Density", main = "Rainfall: Mixture of Normals")  
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "black")  
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "gray")  
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=c("lightgray","black","gray"), lwd = 2)  
  
dev.off()  
  
######################### Helper functions ##############################################