Bayesian Learning - Lab 03

Lakshidaa Saigiridharan (laksa656) and Maximilian Pfundstein (maxpf364)

2019-05-07

Contents

1 Normal Model, Mixture of Normal Model with Semi-Conjugate Prior

Exercise: The data `rainfall.dat` consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of 1 inch, and records of zero 100 precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

a) Assume the daily precipitation $y_1, ..., y_n$ are independent normally distributed, $y_1, ..., y_n|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$

- Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|y_1, ..., y_n)$. The full conditional posteriors are given on the slides from Lecture 7.
- Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.

b) Let us now instead assume that the daily precipitation $y_1, ..., y_n$ follow an iid two-component **mixture of normals** model:

$$p(y_i|\mu, \sigma^2, \pi) = \pi\mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1 - \pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2)$$

where

$$\mu = (\mu_1, \mu_2) \text{ and } \sigma^2 = (\sigma_1^2, \sigma_2^2)$$

Use the Gibbs sampling data augmentation algorithm in `NormalMixtureGibbs.R` (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.

c) Plot the following densities in one figure:

- A histogram or kernel density estimate of the data.
- Normal density $\mathcal{N}(\mu, \sigma^2)$ in (a)

- Mixture of normals density $p(y_i|\mu, \sigma^2, \pi)$ in (b)

Use the posterior mean value for all the parameters.

1.1 Normal Model

First we load the data:

```
################################################################################
# Exercise 1.a)
################################################################################

rainfall_data = read.table("data/rainfall.dat", header=FALSE)

# Prior Parameters (slides page 15)
# These are basically random, guessed or from the data
# mu
mu_0 = 0
tau_sq_0 = 1

# sigma_sq
nu_0 = 1
sigma_sq_0 = 1 # sigma is 1/nu_0
```

```
crinvchisq = function(n, df, s_sq) {
  samples = rchisq(n, df)
  # These are draws from the inverse chi squared
  sigma_sq = (df - 1) * s_sq / samples
  return(sigma_sq)
}
```

```
gibbs_sample = function(nDraws, data, default_sigma, tau_sq_0, mu_0, nu_0, sigma_sq_0) {

  # Posterior Parameters (Taken from lecture 2 slide 4)
  n = length(data)
  mu_n = mean(data) + mu_0
  nu_n = nu_0 + n
  default_sigma_sq = default_sigma^2

  # To store all iterative results
  values_df = data.frame(matrix(NA, nrow = nDraws, ncol = 2))

  # To save current iterative results
  values = list(mu = NaN, sigma_sq = default_sigma_sq)

  # As mu depends on sigma and sigma depends on mu, we need one initial value to start.
  # In this implementation the default value to start with is sigma_sq.

  for (i in 1:nDraws) {
    tau_sq_n = 1 / ((n/values$sigma) + (1/tau_sq_0))
    values$mu = rnorm(1, mu_n, sqrt(tau_sq_n))
    values$sigma_sq = crinvchisq(1, nu_n, (nu_0*sigma_sq_0 + sum((data - values$mu)^2))/(n + nu_0))
    values_df[i,] = values
  }
```

```
  return(values_df)
}
```

```
res = gibbs_sample(nDraws = 1000,
                   data = rainfall_data$V1,
                   default_sigma = 40,
                   tau_sq_0 = tau_sq_0,
                   mu_0 = mu_0,
                   nu_0 = nu_0,
                   sigma_sq_0 = sigma_sq_0)

kable(head(res))
```

X1	X2
32.24222	1570.462
32.53350	1512.809
32.00047	1588.289
32.55267	1568.906
31.60672	1576.676
32.19848	1595.278

```
################################################################################
# Exercise 1.b)
################################################################################
```

1.2 Mixture Normal Model

1.3 Graphical Comparison

2 Metropolis Random Walk for Poisson Regression

Exercise: Consider the following Poisson regression model

$$y_i|\beta \sim \text{Poisson}\left[\exp(x_i^T\beta)\right], i = 1, ..., n$$

where y_i is the count for the ith observation in the sample and x_i is the p-dimensional vector with covariate observations for the ith observation. Use the data set `eBayNumberOfBidderData.dat`. This dataset contains observations from 1000 eBay auctions of coins. The response variable is **nBids** and records the number of bids in each auction. The remaining variables are features/covariates (**x**):

- **Const** (for the intercept)
- **PowerSeller** (is the seller selling large volumes on eBay?)
- **VerifyID** (is the seller verified by eBay?)
- **Sealed** (was the coin sold sealed in never opened envelope?)
- **MinBlem** (did the coin have a minor defect?)
- **MajBlem** (a major defect?)
- **LargNeg** (did the seller get a lot of negative feedback from customers?)
- **LogBook** (logarithm of the coins book value according to expert sellers. Stan- dardized)
- **MinBidShare** (a variable that measures ratio of the minimum selling price (starting price) to the book value. Standardized).

a) Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: `glm.R`, don't forget that `glm()` adds its own intercept so don't input the covariate Const]. Which covariates are significant?

b) Let's now do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim \mathcal{N}[\mathbf{0}, 100 \cdot (X^T X)^{-1}]$ where $\mathbf{X}$ is the $n \times p$ covariate matrix. This is a commonly used prior which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

$$\beta|y \sim \mathcal{N}[\tilde{\beta}, J_y^{-1}(\tilde{\beta})],$$

where $\tilde{\beta}$ is the posterior mode and $J_y(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_y(\tilde{\beta})$ an be obtained by numerical optimization (`optim.R`) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

c) Now, let's simulate from the actual posterior of β using the Metropolis algorithm and compare with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function for any model, I will denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):

$$\theta_p|\theta^{(i-1)} \sim \mathcal{N}(\theta^{(i-1)}, c \cdot \Sigma),$$

where $\Sigma = J_y^{-1}(\tilde{\beta})$ obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis function should be able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across *function objects* in R and the triple dot (...) wildcard argument. I have posted a note (`HowToCodeRWM.pdf`) on the course web page that describes how to do thisin R.

Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.

2.1 Maximum Likelihood Estimator

2.2 Bayesian Analysis of the Poisson Regression

2.3 Simulate from the Actual Posterior Using the Metropolis Algorithm

3 Source Code

```
knitr::opts_chunk$set(echo = TRUE)
library(knitr)

################################################################################
# Exercise 1.a)
################################################################################

rainfall_data = read.table("data/rainfall.dat", header=FALSE)

# Prior Parameters (slides page 15)
# These are basically random, guessed or from the data
```

```r
# mu
mu_0 = 0
tau_sq_0 = 1

# sigma_sq
nu_0 = 1
sigma_sq_0 = 1 # sigma is 1/nu_0


crinvchisq = function(n, df, s_sq) {
  samples = rchisq(n, df)
  # These are draws from the inverse chi squared
  sigma_sq = (df - 1) * s_sq / samples
  return(sigma_sq)
}


gibbs_sample = function(nDraws, data, default_sigma, tau_sq_0, mu_0, nu_0, sigma_sq_0) {

  # Posterior Parameters (Taken from lecture 2 slide 4)
  n = length(data)
  mu_n = mean(data) + mu_0
  nu_n = nu_0 + n
  default_sigma_sq = default_sigma^2

  # To store all iterative results
  values_df = data.frame(matrix(NA, nrow = nDraws, ncol = 2))

  # To save current iterative results
  values = list(mu = NaN, sigma_sq = default_sigma_sq)

  # As mu depends on sigma and sigma depends on mu, we need one initial value to start.
  # In this implementation the default value to start with is sigma_sq.

  for (i in 1:nDraws) {
    tau_sq_n = 1 / ((n/values$sigma) + (1/tau_sq_0))
    values$mu = rnorm(1, mu_n, sqrt(tau_sq_n))
    values$sigma_sq = crinvchisq(1, nu_n, (nu_0*sigma_sq_0 + sum((data - values$mu)^2))/(n + nu_0))
    values_df[i,] = values
  }

  return(values_df)
}


res = gibbs_sample(nDraws = 1000,
                   data = rainfall_data$V1,
                   default_sigma = 40,
                   tau_sq_0 = tau_sq_0,
                   mu_0 = mu_0,
                   nu_0 = nu_0,
                   sigma_sq_0 = sigma_sq_0)
```

```r
kable(head(res))


################################################################################
# Exercise 1.b)
################################################################################
```