Bayesian Learning, 7.5 hp

Home assignment - Part B

Thomas Mortensen, Julia Langenius, Alex Holmskär

2025-09-30

Table of contents

> **Note**
>
> This is the second part of the home assignment for the course. The first part had Problems 1-4, so this second part will continue with this numbering, and the first problem here will therefore be Problem 5.

> **Tip**
>
> It is **strongly advised** to study the notebook for the logistic regression applied to the Titanic data *before* embarking on this assignment. This notebook is here: html | quarto.

Problem 5 - Normal posterior approximation

The file `bugs.csv` contains a dataset with the number of bugs (`nBugs`) and some other explanatory variables for $n = 91$ releases of several software projects. We will use Poisson regression to the model the number of bugs as a function of the following *covariates/features*:

- *intercept* - this is columns of ones to model the intercept
- *nCommit* - the number of commits since the last release of the software
- *propC* - proportion of C/C++ code in the project
- *propJava* - proportion of Java code in the project

- *complexity* - a measure of code complexity that takes into account the frequency of if statements etc.

This code sets up the vector of response observations (`y`) and the 91×5 matrix of covariates `X`

```
data = read.csv("https://github.com/mattiasvillani/BayesLearnCourse/raw/master/assignment/bug
                header = TRUE)
y = data$nBugs # response variable: the number of bugs, a vector with n = 91 observations
X = data[,-1]  # 91 x 5 matrix with covariates
X = as.matrix(X) # X was initially a data frame, but we want it to be matrix
head(X)
```

```
     intercept nCommits     propC  propJava complexity
[1,]         1        5 0.3191181 0.2307104  0.6050054
[2,]         1        4 0.4150649 0.3295860  0.7031914
[3,]         1       13 0.2532424 0.2821941  0.7550588
[4,]         1        1 0.4901135 0.2260926  0.2207819
[5,]         1       10 0.1888998 0.3671323  0.8765245
[6,]         1        7 0.3914173 0.3340347  0.8076927
```

We first consider the **Poisson regression model**

$$Y_i|x_i \overset{\text{ind}}{\sim} \text{Poisson}\Big(\lambda_i = \exp\left(x_i^\top \beta\right)\Big)$$

Note how each observation (software release) has its "own" λ_i parameter, which is modeled as the exponential of $x_i^\top \beta$. The exponential function is used to make sure that λ_i is always positive, as it has to be in the Poisson model.

We ignore here that some of the observations actually come from the same project at different releases, and assume that the response observations Y_i are *independent*, given the features.

The covariates/features for the ith observation $x_i = (x_{1,i}, ..., x_{p,i})^\top$ is the ith row of the matrix `X` . For example, for the second observation we have `y[2] = 6`, so six bugs in the second release, and the covariate values for this second release are:

```
X[2,]
```

```
 intercept   nCommits      propC   propJava complexity
 1.0000000  4.0000000  0.4150649  0.3295860  0.7031914
```

That is, this release (observation) has 4 commits, approximately 41.4% C code, 32.9 % Java code and a Code Complexity of 0.7.

Problem 5a)

Compute a (multivariate) normal approximation of the joint posterior distribution for the vector of Poisson regression coefficients β. Use the prior $\beta \sim N(0, \tau^2 I_p)$ where I_p is the $p \times p$ identity matrix (obtained in R by `diag(5)` when $p = 5$). Set $\tau = 10$ to get a fairly non-informative prior. The library `mvtnorm` contains the multivariate normal density function `dmvnorm`.

Problem 5b)

Plot the (approximate) marginal posterior distribution of each β_j for $j = 1, \ldots, 5$. Summarize each marginal posterior by a 95% interval (you are free to choose the type of interval). As a sort of Bayesian "significance" test, check if the value $\beta_j = 0$ is included in the interval for $j = 1, \ldots, 5$.

> **!** Important
>
> Note that this significance is not the usual frequentist interpretation. There are other Bayesian ways to assess "significance", for example by computing Bayes factors or using Bayesian variable selection. However, the simple version here based on credible intervals is often a more robust choice. In particular, the credible interval approach is less dependent on the prior for β, whereas Bayes factors are known to be very sensitive to the choice of prior and its prior hyperparameters.

Problem 6 - Posterior sampling with the Metropolis-Hastings algorithm

Problem 6a)

Write a function `RWMsampler` that implements the **Random Walk Metropolis algorithm** from Chapter 10 in the Bayesian Learning book. The `RWMsampler` function should have signature

```
RWMsampler <- function(logPostFunc, initVal, nSim, nBurn, Sigma, c, ...){
  # Run the algorithm for nSim iterations
  # using the multivariate proposal N(theta_previous_draw, c*Sigma)
  # Return the posterior draws after discarding nBurn iterations as burn-in
}
```

where `nSim` is the number of draws after the `nBurn` burn-in draws. `logPostFunc` is a function object that computes the log posterior in proportional form and the final triple dot argument … catches all additional arguments (data and the prior hyperparameters) needed to evaluate

the `logPostFunc` function. See the document How to code up a general Metropolis sampler in R for details on all of this.

> **Caution**
>
> The `mvtnorm` package contains a random number generator `rmvnorm` for the multivariate normal distribution. The output of that function is an $n \times p$ matrix where each row is draw of the p-dimensional multivariate normal vector. So when used to generate a single draw $n = 1$, the output is $1 \times p$ matrix, **not** a vector (in R's sense). Use the `as.vector()` function to convert that $1 \times p$ matrix into a vector.

Problem 6b)

Use the RWM sampler from Problem 6a) to sample from the posterior of the Poisson regression model in Problem 5a). Set the proposal covariance matrix Σ equal to the posterior covariance matrix from the Normal approximation in Problem 5) and the RWM scaling constant to $c = 0.5$. Use `nSim = 5000` and `nBurn = 1000`. Use the zero vector as initial value for the algorithm. Plot histograms of the posterior draws to represent the marginal posterior densities.

Problem 6c)

Check if the posterior samples from MCMC seems to have **converged** to the true posterior by:

1. **Plot the MCMC trajectories** (draws over the MCMC iterations) for all parameters.
2. Plot the **cumulative estimates** of the posterior mean for the parameters based on increasing number of draws.
3. **Re-run the sampler** a second time, this time starting from the unit vector $(1, 1, 1, 1, 1)$, and compare the posterior mean estimates from the two runs.

Problem 6d)

Re-run the RWM algorithm, this time using $\Sigma = I_p$ and with $c = 1$ (again using the zero vector as initial value). Is the mixing of the MCMC chain better or worse, compared to the samples obtained in Problem 6b)? Why?

Problem 7 - Posterior sampling with the HMC algorithm in Stan

> **Note**
>
> This exercise uses Probabilistic programming in the Stan language. You need to use the `rstan` and `loo` packages. If you run into troubles installing `rstan` on your own computer, consult the Getting started with Rstan guide.

> **Important**
>
> `rstan` compiles your model the first time you define it. This takes some time. To avoid re-compilation every time you use the model, and to use all available cores on your computer, add the following settings in the beginning of your stan code:

```
library(rstan)
library(loo)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

Problem 7a)

Sample 1000 draws from the posterior distribution of β in the Poisson regression from Problem 5), but this time using the Hamiltonian Monte Carlo algorithm in the `rstan` package.

> **Tip**
>
> Stan includes a special function `poisson_log` that implements the Poisson distribution with a so called *log link*. This is the same as a Poisson distribution with $\lambda = \exp(\theta)$, so that the exponential function (which is the inverse function to the log function) is built-in from the start. Read this: StanUserGuide - Poisson regression.

Problem 7b)

The upcoming release, has the following covariate vector:

```
xNew = c(1, 10, 0.45, 0.5, 0.89)
```

So, the release is based on 10 commits, good proportions of C and Java code and a high code complexity of 0.89. Use Stan to simulate from the predictive distribution of the number of bugs in this release.

> **Tip**
>
> Add a `generated quantities` section to your Stan model to produce the predictive distribution, see Stan User Guide - Prediction.

Problem 7c)

Consider the negative binomial regression for the same bugs data:

$$Y_i|x_i \overset{\text{ind}}{\sim} \text{NegBin}\Big(r, \mu_i = \exp\left(x_i^\top \beta\right)\Big)$$

> **Note**
>
> We are here using the parameterization of the negative binomial distribution where the **second parameter is the mean**; this corresponds to using the `mean` argument in `dnbinom` function in plain R. In Stan, this distribution (again, with the log link built-in) is called `neg_binomial_2_log`, see Stan User Guide - negative binomial regression.

The parameters in this model are therefore the vector of negative binomial regression coefficients β **and** the scalar parameter $r > 0$. Use HMC in Stan to sample 1000 draws from joint posterior $p(\beta, r|y, X)$. Plot the marginal posterior for r. What does this posterior tell you about the suitability of the Poisson regression model in 7a)?

Problem 7d)

Compare the Poisson regression and Negative binomial regression models using Bayesian **leave-one-out (LOO) cross-validation**. Use the posterior samples from `rstan` and the `loo` package to compute the expected log predictive density from leave-one-out cross-validation (**ELPD-LOO**) for each model. Which model is preferred?

> **Note**
>
> Computing the ELPD-LOO from the Stan output requires that we tell Stan to store the log-likelihood values for each HMC draw. Similar to prediction, we achieve this by adding the log-likelihood computation to the `generated quantities` section. See the Writing Stan programs for use with the loo package.