

Bayesian Learning

Lab 4

Emil K Svensson and Rasmus Holm

2017-05-24

Question 1

```
bid <- read.table("../data/eBayNumberOfBidderData.dat", header = TRUE)
```

a)

```
glm_res <- glm(nBids ~ . - 1, data = bid, family = poisson(link = "log"))
summary(glm_res)
```

```
##
## Call:
## glm(formula = nBids ~ . - 1, family = poisson(link = "log"),
##      data = bid)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.580  -0.722  -0.044   0.527   2.461
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## Const           1.0724    0.0308   34.85 < 2e-16 ***
## PowerSeller    -0.0205    0.0368   -0.56  0.577
## VerifyID       -0.3945    0.0924  -4.27 2.0e-05 ***
## Sealed          0.4438    0.0506   8.78 < 2e-16 ***
## Minblem        -0.0522    0.0602  -0.87  0.386
## MajBlem        -0.2209    0.0914  -2.42  0.016 *
## LargNeg         0.0707    0.0563   1.25  0.210
## LogBook        -0.1207    0.0290  -4.17 3.1e-05 ***
## MinBidShare    -1.8941    0.0712 -26.59 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##      Null deviance: 6264.01  on 1000  degrees of freedom
## Residual deviance:  867.47  on  991  degrees of freedom
## AIC: 3610
##
## Number of Fisher Scoring iterations: 5
```

b)

```
library(mvtnorm)

logprior <- function(beta, mu, sigma){
  dmvnorm(beta, mean = mu, sigma = sigma, log = TRUE)
}

loglikelihood <- function(beta, X, Y){
  linear_prediction <- t(X) %*% beta
  probabilities <- Y * linear_prediction - exp(linear_prediction)
  loglike <- sum(probabilities)

  ## if (abs(loglike) == Inf)
  ##   loglike = -20000

  loglike
}

## loglikelihood <- function(beta, X, Y) {
##   linear_prediction <- t(X) %*% beta
##   probs <- dpois(Y, lambda = exp(linear_prediction), log = TRUE)
##   sum(probs)
## }

logposterior <- function(beta, X, Y, prior_mu, prior_sigma){
  loglikelihood(beta, X, Y) + logprior(beta, prior_mu, prior_sigma)
}

X <- as.matrix(bid[,-1])
Y <- as.matrix(bid[,1])

mu <- rep(0, ncol(X))
sigma <- 100 * solve(t(X) %*% X)

normal_res <- optim(par = matrix(rep(0, ncol(X)), ncol = 1),
  fn = logposterior, method = "BFGS", hessian = TRUE,
  X = t(X), Y = Y,
  prior_mu = mu, prior_sigma = sigma,
  control = list(fnscale = -1))
hessian <- normal_res$hessian
```

c)

```
targetdensity <- function(theta, prior_mu, prior_sigma, X, Y, ...) {
  likelihood <- dpois(Y, lambda = exp(t(X) %*% t(theta)), log = TRUE)
  prior <- dmvnorm(theta, mean = prior_mu, sigma = prior_sigma, log = TRUE)
  sum(likelihood) + prior
}

proposaldensity <- function(x, mu, prop_sigma, ...){
  dmvnorm(x, mean = mu, sigma = prop_sigma, log = TRUE)
}
```

```

}

proposalsampler <- function(mu, prop_sigma, ...){
  matrix(rmvnorm(1, mean = mu, sigma = prop_sigma), nrow = 1)
}

metropolis_hastings <- function(prop_sampler, log_prop_func, log_targ_post_func,
                                X0, iters, ...){
  x <- X0
  values <- matrix(0, ncol = length(X0), nrow = iters + 1)
  values[1,] <- X0

  alpha <- function(x, y, ...) {
    numerator <- log_targ_post_func(y, ...) + log_prop_func(x, y, ...)
    denominator <- log_targ_post_func(x, ...) + log_prop_func(y, x, ...)
    exp(numerator - denominator)
  }

  for (i in 1:iters) {
    y <- prop_sampler(x, ...)
    u <- runif(1)

    if (u < alpha(x, y, ...)) {
      x <- y
    }

    values[i+1,] <- x
  }

  values
}

iters <- 5000
X0 <- rep(0, times = ncol(X))

params <- list(
  prop_sampler = proposalsampler,
  log_prop_func = proposaldensity,
  log_targ_post_func = targetdensity,
  X0 = matrix(rep(0, times = ncol(X)), nrow = 1),
  iters = iters,
  X = t(X),
  Y = Y,
  prior_mu = rep(0, times = ncol(X)),
  prior_sigma = 100 * solve(t(X) %*% X),
  prop_sigma = 0.6 * -solve(hessian)
)

metro_res <- do.call(metropolis_hastings, params)

as.vector(normal_res$par)

```

```
## [1] 1.0698 -0.0205 -0.3930 0.4436 -0.0525 -0.2212 0.0707 -0.1202 -1.8920
```

```
as.vector(coef(glm_res))
```

```
## [1] 1.0724 -0.0205 -0.3945 0.4438 -0.0522 -0.2209 0.0707 -0.1207 -1.8941
```

```
as.vector(colMeans(metro_res))
```

```
## [1] 1.0302 -0.0223 -0.3783 0.4233 -0.0615 -0.1956 0.0796 -0.1201 -1.8483
```

d

```
Xpred <- matrix(c(1, 1, 1, 1, 0, 0, 0, 1, 0.5), nrow = 1)  
predsamples <- rpois(10000, lambda = exp(Xpred %*% t(metro_res)))
```

```
mean(predsamples == 0)
```

```
## [1] 0.362
```

```
hist(predsamples, breaks = 50)
```

Histogram of predsamples

