STAT 580 Homework 5

Yifan Zhu

April 4, 2017

1. (a) See part (c).

(b) $f(\lambda|p, \boldsymbol{r}, \boldsymbol{x}) \propto \lambda^{a-1}\mathrm{e}^{-b\lambda}\prod_{i=1}^n \mathrm{e}^{-\lambda r_i}\lambda^{x_i} = \lambda^{a+\sum_{i=1}^n x_i - 1}\mathrm{e}^{-\lambda(\sum_{i=1}^n r_i + b)}$. Hence

$$(\lambda|p, \boldsymbol{r}, \boldsymbol{x}) \sim \text{Gamma}(a + \sum_{i=1}^n x_i, b + \sum_{i=1}^n r_i)$$

.

$f(p|\lambda, \boldsymbol{r}, \boldsymbol{x}) \propto \prod_{i=1}^n p^{r_i}(1-p)^{1-r_i} = p^{1+\sum_{i=1}^n r_i - 1}(1-p)^{n+1-\sum_{i=1}^n r_i} - 1$. Hence

$$(p|\lambda, \boldsymbol{r}, \boldsymbol{x}) \sim \text{Beta}(1 + \sum_{i=1}^n r_i, n + 1 - \sum_{i=1}^n r_i)$$

.

$f(r_i|\lambda, p, \boldsymbol{x}) = C\mathrm{e}^{-\lambda r_i} r_i^{x_i} p^{r_i}(1-p)^{1-r_i}$. When $x_i = 0$, $f(1|\lambda, p, \boldsymbol{x}) = Cp\mathrm{e}^{-\lambda}$, $f(0|\lambda, p, \boldsymbol{x}) = C(1-p)$; when $x_i \neq 0$, $f(1|\lambda, p, \boldsymbol{x}) = Cp\mathrm{e}^{-\lambda}$, $f(0|\lambda, \pi, \boldsymbol{x}) = 0$. Hence

$$(r_i|\lambda, p, \boldsymbol{x}) \sim \text{Bernoulli}(\frac{p\mathrm{e}^{-\lambda}}{p\mathrm{e}^{-\lambda} + (1-p)I(x_i = 0)})$$

.

(c)

```r
# (a) Generate X = RY

library(Rlab)
x <- rbern(n = 100, p = 0.3) * rpois(n = 100, lambda = 2)

# (c) Simulation using Gibbs Sampler

# Initialize lambda and p
lambda <- 1
p <- 0.5

# vector to store simulated lambda's and p's
lambda_chain <- array(dim = 5000)
p_chain <- array(dim = 5000)

# Gibbs Sampling
a <- 1
b <- 1
n <- length(x)
Ix <- ifelse(x==0,1,0)
rx <- 1 - Ix
Sx <- sum(x)
for (i in 1:5000){
```

```r
  r <- rbern(n = 100, p = (p*exp(-lambda))/(p*exp(-lambda) + (1 - p)
     )) * Ix + rx
  lambda <- rgamma(1, shape = a + Sx, rate = b + sum(r))
  lambda_chain[i] <- lambda
  p <- rbeta(1, shape1 = 1 + sum(r), shape2 = n + 1 - sum(r))
  p_chain[i] <- p
}

# drop the first 1000 lambda's and p's
lambda_chain <- lambda_chain[1001:5000]
p_chain <- p_chain[1001:5000]

quantile(lambda_chain, c(0.025, 0.975))
quantile(p_chain, c(0.025,0.975))
```

CI for λ is $(1.495806, 2.579958) \ni 2$. CI for p is $(0.2636606, 0.4808380) \ni 0.3$.

2.

```r
# Creat vector for storing Z's
Z_chain <- array(dim = 1500)

# Initialize Z0
Z <- 1

# parameters for gamma distribution
a <- 1
b <- 1

# parameters for Z
theta1 <- 1.5
theta2 <- 2

for (i in 1:1500){
  Y <- rgamma(1, shape = a, rate = b)
  U <- runif(1)
  r <- (Z/Y)^(3/2)*exp(theta1*(Z - Y) + theta2*(1/Z - 1/Y)) * (Z/Y)^(a -
      1)*exp(b*(Y - Z))
  if (U <= r){
    Z <- Y
  }
  Z_chain[i] <- Z
}

# drop the first 500 simulations
Z_chain <- Z_chain[501:1500]

# true values
EZ <- sqrt(theta2/theta1)
EZinverse <- sqrt(theta1/theta2) + 1/(2*theta2)

# estimates
EZ_sim <- mean(Z_chain)
EZinverse_sim <- mean(1/Z_chain)

cbind(EZ, EZinverse, EZ_sim, EZinverse_sim)
```

True values: $E\left(Z\right)_{\text{true}} = 1.154701$ and $E\left(\frac{1}{Z}\right)_{\text{true}} = 1.116025$.

Estimated values: $E\left(Z\right)_{\text{estimate}} = 1.103276$ and $E\left(\frac{1}{Z}\right)_{\text{estimate}} = 1.162151$

3. Call from R:

```r
# Generate X = RY

library(Rlab)
x <- rbern(n = 100, p = 0.3) * rpois(n = 100, lambda = 2)


# parameter used for Gibbs Sampling
a <- 1
b <- 1

dyn.load("~/Desktop/Homework/STAT580/STAT580hw5YifanZhu/Codes/Gibbs.so")

samples <- .Call("Gibbs", a, b, x)
```

```
# CI for lambda
quantile(samples[[1]], c(0.025, 0.975))

# CI  for p
quantile(samples[[2]], c(0.025, 0.975))
```

Gibbs Sampling function in C:

```
#include <R.h>
#include <Rinternals.h>
#include <Rmath.h>

SEXP Gibbs(SEXP Ra, SEXP Rb, SEXP Rx){
    int a = asReal(Ra), b = asReal(Rb), n = length(Rx), i, j;
    int *x = INTEGER(Rx);
    int r[n];

    SEXP lambdas, ps;
    PROTECT(lambdas = allocVector(REALSXP, 4000));
    PROTECT(ps = allocVector(REALSXP, 4000));
    double *lambda_chain = REAL(lambdas);
    double *p_chain = REAL(ps);



    // Initiate lambda and p

    double lambda = 1, p = 0.5;

    double Sx = 0, Sr;

    for (i = 0; i < n; i++)
        Sx = Sx + x[i];

    GetRNGstate();
    // burn-in
    for (i = 0; i < 1000; i++){
        Sr = 0;
        for (j = 0; j < n; j++){
            if (x[j] == 0){
                r[j] = rbinom(1, p * exp(-lambda))/(p*exp(-lambda) + (1
                    - p));
            }
            else
                r[j] = 1;
            Sr = Sr +r[j];
        }


        lambda = rgamma(a + Sx, 1/(b + Sr));

        p = rbeta(1 + Sr, n + 1 - Sr);
    }


    // take sample from p's and lambda's for every step
    for (i = 0; i < 4000; i++){
```

```
        Sr = 0;
        for (j = 0; j < n; j++){
            if (x[j] == 0){
                r[j] = rbinom(1, p * exp(-lambda))/(p*exp(-lambda) + (1
                    - p));
            }
            else
                r[j] = 1;
            Sr = Sr +r[j];
        }

        lambda = rgamma(a + Sx, 1/(b + Sr));

        lambda_chain[i] = lambda;

        p = rbeta(1 + Sr, n + 1 - Sr);

        p_chain[i] = p;
    }

    PutRNGstate();

    SEXP samples;
    PROTECT(samples = allocVector(VECSXP, 2));
    SET_VECTOR_ELT(samples, 0, lambdas);
    SET_VECTOR_ELT(samples, 1, ps);

    UNPROTECT(3);
    return samples;
}
```

CI for λ is $(1.566422, 2.690176) \ni 2$. CI for p is $(0.2198559, 0.4139339) \ni 0.3$.