

Gibbs Sampling

Margarita Orlova, Kenneth Ruiter, Patrick Walker

3 november 2018

Gibbs Sampling

For this part of the report we use Gibbs Sampling as a means to estimate the marginal distribution generated from conditional distributions. These distributions are given by

$$p(x|y) \propto ye^{-yx}, \quad 0 < x < B < \infty$$

$$p(y|x) \propto xe^{-yx}, \quad 0 < y < B < \infty$$

where B is a known positive constant. To use Gibbs Sampling, we first need to be able to draw from these conditional distributions. We can use Inverse Transform Sampling to generate these samples, however we first need to know the inverse of the cdf. Therefore we will start by computing this function.

Inverse Transform Sampling

Since the densities are symmetric, we can compute just the inverse of the cdf of X given Y , and then deduce from that the inverse of the cdf of Y given X . We can write the pdf as follows

$$p(x|y) = cye^{-yx}, \quad 0 < x < B < \infty$$

for some constant c , so that the density function integrates to 1. The cdf is then given by

$$F_{X|Y}(x|y) = \int_0^x p(s|y) \, ds = \int_0^x cye^{-ys} \, ds = c[-e^{-ys}]_{s=0}^x = c(1 - e^{-xy})$$

Since we want this to integrate to 1, we now know that $c = \frac{1}{1-e^{-By}}$, as $0 < x < B$, so that we now have our conditional cdf of X given Y . We can then find the inverse of the cdf, $F_{X|Y}^{-1}(x|y)$, by solving the following equation.

$$x = \frac{1}{1 - e^{-By}} (1 - e^{-F_{X|Y}^{-1}(x|y) \cdot y}) \implies F_{X|Y}^{-1}(x|y) = -\frac{1}{y} \log(1 - x(1 - e^{-By}))$$

So now that we have found the inverse of the cdf, we can generate a random draw from this conditional distribution by generating a random draw from the uniform distribution U from 0 to 1, and then compute $F_{X|Y}^{-1}(U|y)$. Similarly, we can draw a random sample from the conditional distribution for Y given X , by drawing a different U uniformly, and then computing $F_{Y|X}^{-1}(U|x) = F_{X|Y}^{-1}(U|x)$.

Algorithm Description

Now that we can sample from the given conditional distributions, we will explain how to use Gibbs sampling to estimate the marginal distributions. This algorithm takes as inputs the number of iterations, the number to thin by (explained later) and the value of B . As for the algorithm itself, first 2 starting values for x and y will be randomly drawn. Then we will iteratively draw values from the conditional distributions, and only store the values of x and y every so often. How often we do this is decided by the set number to thin by. We repeat this a number of iterations, and finally output the stored values of x and y . We can look at just the x values, from which we can find our estimate for the marginal density of x , and vice versa for y . The code for the algorithm is given below.

```

import math
import random
import numpy
import matplotlib as plt
import seaborn as sns
import statistics

sns.set(color_codes = True)

def gibbs(N = 50000, thin = 1000, B = 5):
    mat = numpy.empty((N + 1, 3,)) #create an empty numpy matrix
    B=5 #assigning upper bound
    x = numpy.random.uniform(0,B) #pick x randomly from 0 to 5
    y = numpy.random.uniform(0,B) #pick y randomly from 0 to 5
    mat[0,0] = 0 #setting an iteration number equal to 0 for starting values
    mat[0,1] = x #storing starting value x in matrix
    mat[0,2] = y #storing starting value y in matrix

    for i in range(1,N + 1):

        u1 = random.uniform(0,1) #draw an u value that will be used in inverse sampling
        newx = -(1/y)*math.log(1-u1*(1 - math.exp(-B*y)))
                #inverse CDF of x that was calculated manually before

        x = newx #assign calculated newx as x

        u2 = random.uniform(0,1) #draw an u value that will be used in inverse sampling
        newy = -(1/x)*math.log(1-u2*(1 - math.exp(-B*x)))
                #inverse CDF of y that was calculated manually before

        y = newy #assign calculated newy as y

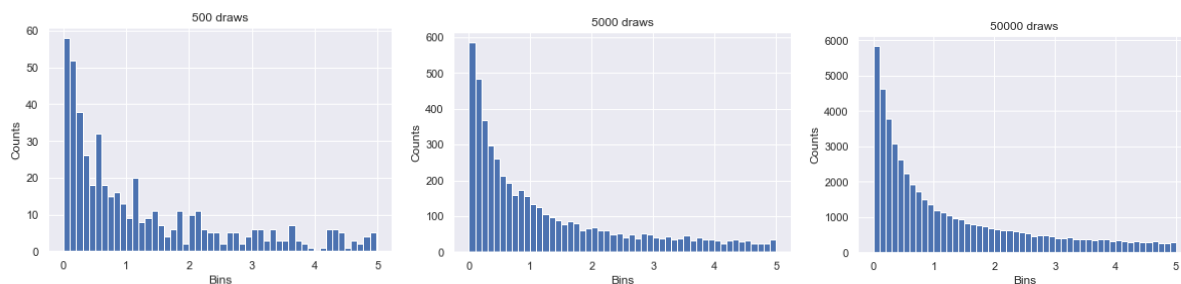
        mat[i,0] = i #number row
        mat[i,1] = newx #store newx in a matrix in row i
        mat[i,2] = newy #store newy in a matrix in row i

    mat = numpy.matrix(mat)
    return(mat) #return numpy matrix, where first column is iteration number,
                #second is x and third in y

```

Results

For the following results, we use that $B = 5$, and create histograms of values for x , for a various amount of iterations T .



To create these plots, the following code was used.

```
trials = gibbs(500) #Example for 500 iterations.  
    #Other plots are created using 5000 and 50000 iterations  
plt.pyplot.hist(trials[:,1],bins = 50) #Create histogram  
plt.pyplot.title('500 draws') #Add title  
plt.pyplot.xlabel('Bins') #Add label on x-axis  
plt.pyplot.ylabel('Counts') #Add label on y-axis
```

As we can see from the three plots shown above, the density of x seems to have a similar shape for all amounts of iterations, where more iterations result in a smoother empirical density. Remember that the values of x will always be in between 0 and $B = 5$, so these histograms show all values of x . Now aside from an estimation of the marginal densities found by the histograms, we can also use Gibbs sampling to find estimates for the moments of these marginal densities. For example, we know that the sample average is an unbiased and consistent estimator for the first moment of X , $\mathbb{E}_{p(X)}[X]$. We can simply take the mean of all x -values that are given in the output to find this sample average. The following code was used.

```
statistics.mean(numpy.asarray(trials)[:,1])  
    #The same trials were used for creating the histograms,  
    #so the following averages correspond with these histograms.  
    #Again, this is only the example for 500 iterations.
```

The estimates for the first moments are then given below.

- 500 iterations: $\mathbb{E}_{p(X)}[X] \approx 1.2841215406495794$
- 5000 iterations: $\mathbb{E}_{p(X)}[X] \approx 1.2586229563432523$
- 50000 iterations: $\mathbb{E}_{p(X)}[X] \approx 1.2697125546442287$