A second look at Implementation of Gibbs Sampling within Bayesian Inference and its Applications in Actuarial Science

Colton Gearhart

Abstract

This report discusses one use of a Bayesian approach within an actuarial context. Specifically, we will investigate a particular Markov chain Monte Carlo method, Gibbs sampling. This technique enables complicated actuarial models to be analyzed by reducing them to simpler models. The properties of Gibbs sampling are demonstrated and then applied within SAS. The implementation of this algorithm involved several macros, which utilized a variety of programming concepts. Ultimately, we were able to estimate a density that does not have a closed form solution.

Introduction

Simulation is an important tool for actuaries when they are attempting to study stochastic models. Often, actuarial models are quite complicated, if not impossible to find closed form solutions for. If this is the case, simulation becomes a valuable resource. When we combine Monte Carlo simulation techniques with Bayesian analysis, it enables us to learn about the entire distribution of a quantity. In addition, this approach allows for modelling uncertainty by assigning probability distributions to parameters, not just the data. This is worked into our models via the prior distributions of our parameters, which are then updated with data. In effect, as the simulation continues, we are supplying more and more information to our parameters. This ultimately increases reliability of our estimates. Finally, in order to estimate the target integrals that we could not sample from directly, we can apply discrete formulas to our sample.

One popular method of performing this process is Gibbs sampling. Rather than sampling from the complicated multivariate posterior distributions (which may not even be possible), Gibbs sampling generates random draws from the full conditional distributions. After each draw is made, the conditional distributions of the variables get updated with the new information. This is shown by the basic Gibbs sampling algorithm shown below (*j* represents the iteration count):

```
0) $x^{(j)} = \left(x_1^{(j)}, x_2^{(j)}, \dots, x_k^{(j)}\right)$, with $j = 0$.
1) Set $j = j + 1$.
2) Simulate $x_1^{(j)} \sim f\left(x_1 | \, x_2^{(j-1)}, x_3^{(j-1)}, \dots, x_k^{(j-1)}\right)$.
3) Simulate $x_2^{(j)} \sim f\left(x_2 | \, x_1^{(j-1)}, x_3^{(j-1)}, \dots, x_k^{(j-1)}\right)$.
…  …
```

k) Simulate $x_k^{(j)} \sim f\left(x_k \mid x_1^{(j-1)}, x_2^{(j-1)}, \dots, x_{k-1}^{(j-1)}\right)$.

k+1) Form $x^{(j)} = \left(x_1^{(j)}, x_2^{(j)}, \dots, x_k^{(j)}\right)$.

k+2) Return to step 1.

In essence, each draw depends on the previous one. In addition, with other parameters being set at their current values (i.e. they are treated as fixed), these joint densities often simplify to known distributions. This result makes many problems much easier.

The goal of this project is to investigate the properties of Gibbs sampling, a Markov chain Monte Carlo method, in addition to applying it on an example. Specifically, we will look at how a basic Gibbs sampling algorithm can be implemented within SAS.

Methods

Simulate

In order to demonstrate the properties of Gibbs sampling, we will look at an example involving the size of claims. We start by assuming the size of a claim *X* is exponentially distributed given rate parameter λ. Additionally, let λ be a random variable following a gamma distribution with parameters α and β, both of which are constants. After deriving the conditional distribution for $\lambda|x$, we find that it follows a gamma distribution with parameters α+1 and x+β. We can now implement a Gibbs sampling algorithm in SAS.

In order to do this, we first set the number of random numbers to generate (*n*), the number of burn-in iterations to be used (*burnin*), and values of parameters as macro variables (*alpha, beta*). Burn in iterations are used to allow the Markov chains to converge. Next, we initialized starting values of *x*, λ and the counter variable within a data step. Then a do loop generated *n* random numbers for *x* and λ from the dependent sampling scheme defined in the introduction. Lastly, we removed the burn-in iterations using an if statement. With the data simulated, scatter plots of the last 100 generated values for *x* and λ were created using *proc sgplot*. Histograms of the simulated data were also created using *proc sgplot*.

Estimate

We can now approximate the desired integrals by applying discrete formulas to our samples. For our marginal distribution of interest $f(x) = \int f(x|\lambda)f(\lambda)d\lambda$, we can obtain an estimate of the actual value *f(x)* at point *x* using $f(\hat{x}) = \frac{1}{n-burnin} \sum_{i=burnin}^{n} f(x|\lambda_i)$. If we do this process over many values in the range of *X*, we can obtain an estimated density of *f(x)*. Similar calculations can be done for that of λ. To do this in SAS, macros were utilized. The *estimate* macro takes the following arguments as parameters: the simulated dataset, the name of the output dataset, variables to estimate, the number of variables to estimate, and information about which values to estimate each variable at. For this example, here is how the macro works. Within a data step, it initializes the starting value of the random variable *X* and creates an array with a different variable for each value *x* that we are going to estimate at. Next, it uses a do loop to create a dataset where each observation is calculated according to $f(x|\lambda_i)$. It then utilizes *proc means* to create a wide dataset of the final density estimates of *X* at point *x* by averaging over all of the λs.

In order to get the data to a more manageable form, another macro is used. The *wideToLong* macro takes the following arguments as parameters: the wide dataset, the long dataset to be created, the variables to stack, an indicator to keep the row index or not, and which additional variables to keep from the wide dataset. Upon being called, this macro returns the desired reshaped dataset. Finally, the last step in the *estimate* macro is to combine final density estimates of *X* with the corresponding inputs.

Iterative do statements, conditionally processing, and the technique of resolving macro variables allow this entire process to be repeated for each of the variables specified when calling the *estimate* macro. So, for this example, calling this macro resulted in two datasets of estimated densities, one for *X* and one for λ. This macro was designed to be very generalized and can be used again for the later example. There are some minor cavoites in terms of the generalization though. The conditional distributions used in the intermediate calculations need to be manually edited to fit the given situation; additionally, parameters that correspond to information about which values to estimate at may need to be added or taken away based on the number of random variables in the given situation. This macro is also efficient in terms of created datasets as there is only one dataset output for each random variable (all intermediate datasets are removed from the work library using *proc datasets*).

Validate

In practice, the marginal distributions of interest often cannot be solved for in closed form, which is why Gibbs sampling is of use. However, for this example, we chose distributions such that there would be closed-form solution so that our results can be compared. We know that $X \sim Pareto(\alpha, \beta)$ and $\lambda \sim gamma(\alpha, \beta)$. In order to plot these marginal densities with *proc sgplot*, we had to create datasets of the pdf values at many input values. Then the *series* statement in *proc sgplot* can interpolate along these points to form the marginal density curves. These were overlaid on the same plots of the estimated densities of *X* and λ, respectively.

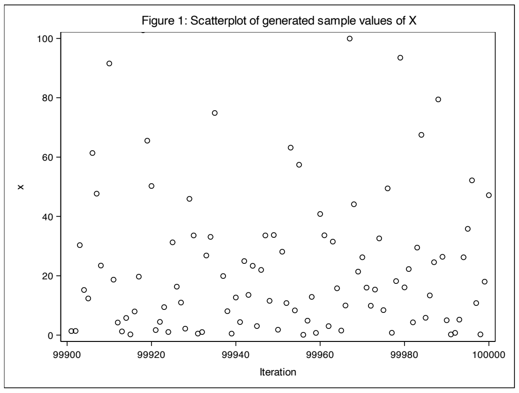
Apply

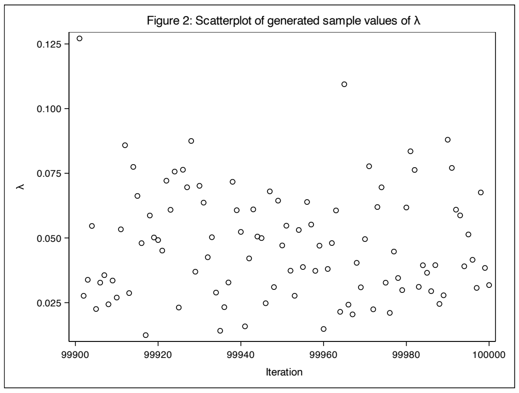
Utilizing the *estimate* macro, we could very easily perform another Gibbs sampling algorithm. Consider the discrete situation where we would like to model number of claims. Let the probability of filing a claim *P* follow a beta distribution with parameters α and β. Additionally, let the number of policies in a given portfolio *N* follow a zero-truncated Poisson distribution with parameter λ. Finally, let the number of policies that file a claim follow a binomial distribution with parameters *N* and *P*. In this scenario, *f(x)* cannot be solved for in closed form. So, in order to estimate it, we will utilize Gibbs sampling.

Using the same code from the prior example, only a few changes needed to be made to fit this scenario. The simulated random numbers needed to come from conditional binomial, beta and Poisson distributions, respectively. Then, because we are interested in the marginal density of *X*, the typical number of policies that generated a claim within an arbitrary portfolio, we needed to change the conditional distribution that SAS will average over. After doing this and initializing the macro with the correct parameters, we called the *estimate* macro. This resulted in a density estimate for the marginal distribution of *X*.

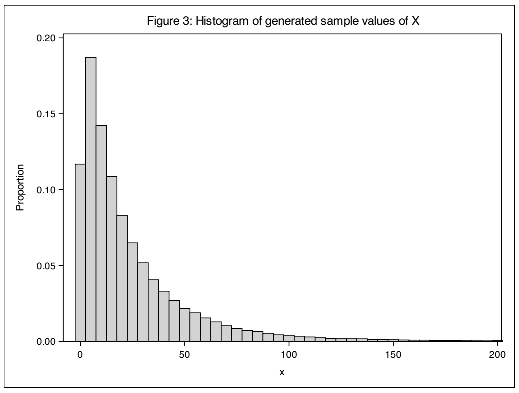
Results

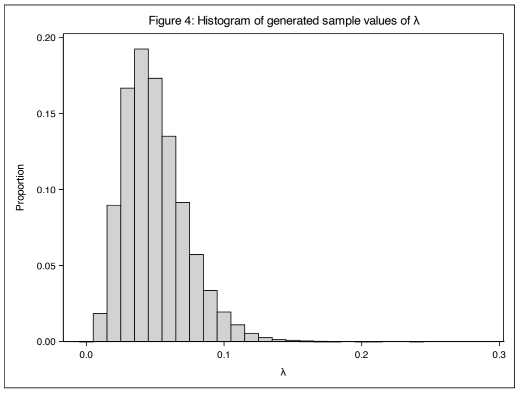
After simulating the random numbers for the first example, we wanted to check and make sure that they appeared to be a random sample. Scatter plots of the last 100 generated values of *X* and λ, respectively, were created to check this. These are shown below in Figures 1 and 2.



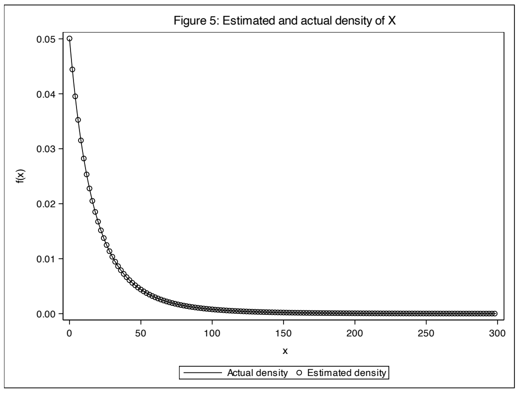


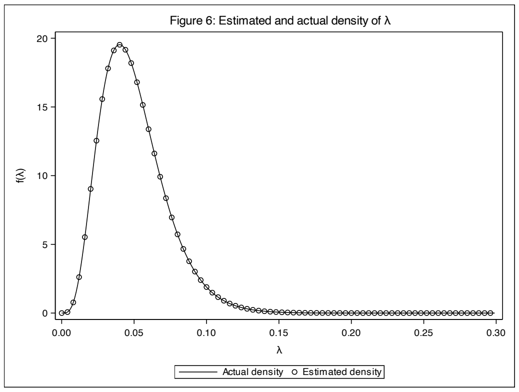
These plots indicate that there is no pattern among the generated random numbers. Thus, we can consider them to be independent random samples. Next, we created histograms of the generated samples to look at the overall distributions of each. These are shown in Figures 3 and 4.





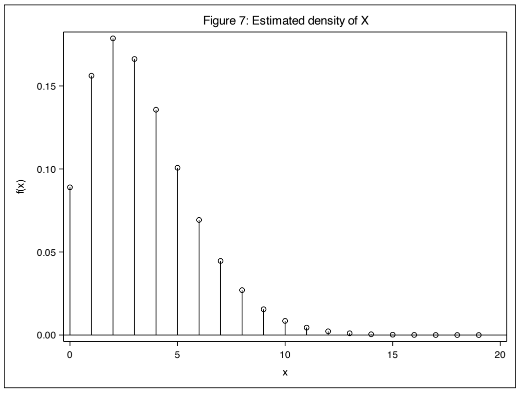
Finally, plots of the estimated densities of *X* and λ with their respective actual marginal densities overlaid were created. These are shown in Figures 5 and 6.





It appears that the respective marginal densities line up very well with their estimated densities. If looking at more complicated models, these results suggest that we can use Gibbs sampling to study random samples from unknown distributions.

This process was then applied to a different situation in which we wanted to model the number of policies that file a claim within an arbitrary portfolio. The estimated density of this is shown in Figure 7.



With these results, actuaries can compute several statistics for a portfolio with parameters *p* and *n*, which could ultimately help improve their predictions for the expected number of future claims.

Conclusion

Overall, this project had the goal of demonstrating the properties of Gibbs sampling and providing a possible application. Specifically, we wanted to show how this could be implemented within SAS. Future work could include investigating how many burn-in iterations are necessary for the Markov chains to converge or why the conditional distributions only need to be known up to a normalizing constant. One remaining question that we have relates to the idea of convergence; we are not sure exactly how this works. For example, how are the generated numbers “less random” after many iterations? We suspect this knowledge to come to us in the upcoming semesters.

References

Gearhart, C., Implementation of Gibbs Sampling within Bayesian Inference and its Applications in Actuarial Science, SIAM Undergraduate Research Online, Vol. 11.

SAS Institute Inc. 2015. SAS/IML® 14.1 User’s Guide. Cary, NC: SAS Institute Inc.