ST2195 COURSEWORK

Report for Part 1

SRN:
Total number of pages: 2 (including TOC)

Table of Contents

1. Metropolis-Hastings Algorithm *2*

1.1 Applying the Random Walk Metropolis Algorithm **2**

1.2 Calculating R **2**

1. Metropolis-Hastings Algorithm

1.1 Applying the Random Walk Metropolis Algorithm

In this section, similar graphs for R and Python shows the Histogram, Kernel Density Plot, and the Probability Density Function, which was generated using the Random Walk Metropolis Algorithm. It has a title of 'Histogram of generated values', y-axis of 'Density' and x-axis of 'values of x'. Figure 1.1a shows the graph on R, and Figure 1.1b shows the graph on Python. Both holds a general shape of a normal distribution graph. This was generated using s=1, and an iteration of N=10000. It has a mean of 0.048 and a standard deviation of 1.339 for R, and similar values of -0.002 and 1.420 respectively for Python. The Probability Density Function peaks at Density=0.5, while the Kernel Density Plot slightly falls under that.

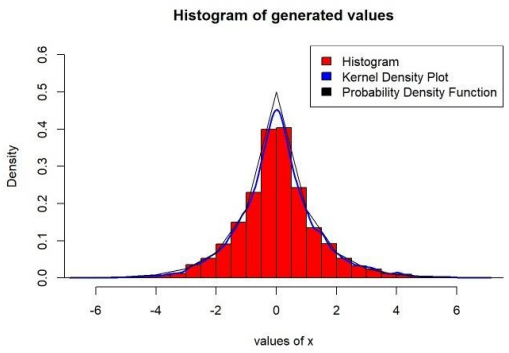

Figure 1.1a Random Walk Metropolis (R)

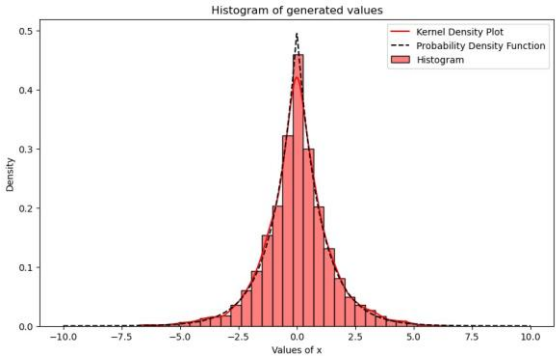

Figure 1.1b Random Walk Metropolis (Python)

1.2 Calculating $\hat{R}$

$\hat{R}$ is a convergence diagnostic value, where values of $\hat{R}$ close to 1 indicates convergence. Firstly, s=0.001 was used to calculate $\hat{R}$. Following the calculations provided in the question, we will generate a $\hat{R}$ value of 58.09 for R, and 28.12 for Python. This suggest that the algorithm had not converged when s=0.001. Next, we will use s values in the interval of 0.001 to 1 to calculate a series of $\hat{R}$ values.

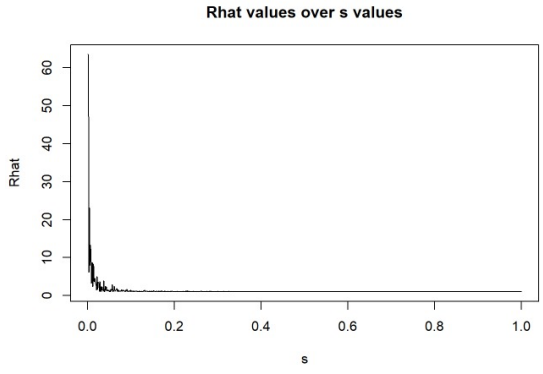

Figure 1.2a Rhat (R)

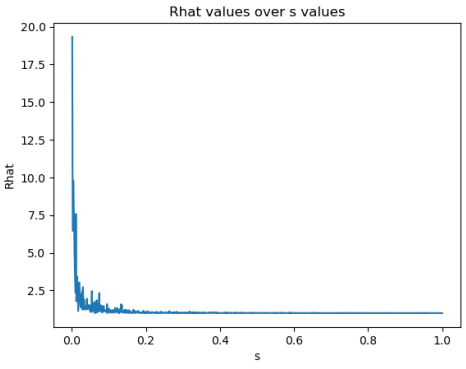

Figure1.2b Rhat (Python)

The figures above show the graph of $\hat{R}$ values over a grid of s values in the interval between 0.001 to 1, with a y-axis of 'Rhat' and x-axis of 's'. Figure 1.2a shows the graph in R, and Figure 1.2b shows the graph in Python. Both graphs have a downward sloping curve, with jagged areas in the 'elbow' area. They are the most prominent from s values of 0 to 0.3. It also can be seen that $\hat{R}$ is always positive, and that as the value of s increases, $\hat{R}$ value will decrease. Initially, the algorithm had not converged as the $\hat{R}$ value is more than 1. However, after s value = 0.3, the $\hat{R}$ value mainly stabilises to about 1, which shows that the algorithm had converged.