STAT 243 Project

Github: https://github.com/caojilin/stat243project

December 2020

This document will present the approach proposed by Jilin Cao, Jason Nyam and Tianchen Xu to implement an adaptive-rejection sampler. We will describe in this document the different steps in the approach, the functions used and the choices made for the implementation. Our solution is based on Section 2.2 of Gilks et al. (1992).

1 Initialisation

This step consists in initializing the different variables and checking the validity of the inputs. We created the function "ars" that takes as inputs: - n the number of desired samples
- g the density function of interest
-D_left The desired left end of domain (optional)
-D_right The desired right end of domain (optional)
-k The number of desired initial Abscissa (optional)
We insure that k, D_right, D_left and n are numbers, g is a function and D_right > D_left using "check_param"

Then we initialize the variables: we compute h and h' with $h(x) = log(g(x))$ and check that h is concave by looking at the monotony of h'. We initialize T_k in "init_T_k" such as $T_k = \{x_i, i = 1, ..., k\}$, where $x_1 \leq \ldots \leq x_k$ are the k abscissa in D where we will evaluate $h(x)$ and $h('x)$ If D unbounded on the left, we chose x_1 such that $h'(x1) > 0$ If D unbounded on the right, we chose x_k such that $h'(x_k) < 0$

Then we created two vector with the values of $h(x_i)$ and $h'(x_i)$, we uses those vectors in "calc_z" to compute $z_1 \ldots z_k$ with the formula given in Gilks.

Finally for every x_i in T_k we compute $s_k(x_i)$ with the formula in Gilks.

2 Sampling

In this step, we sample a new x^* from s_k. We used the distr package and sample x^* from s_k using the functions r and AbscontDistribution.

Once we have x^*, we need to compute $l_k(x^*)$ and $u_k(x*)$. We created functions "calc_l_j" "calc_u_j" that computes $l_k(x)$ and $u_k(x)$ with the formula from Gilks for a given x, but we need to know in which interval of z for u_k and of T_k

for l_k x^* is in. So we used the findInterval function and set $l_k(x) = -\inf$ if x^* is out of T_k.

We also need to decide tested whether to accept or reject x^*. We sample w from a uniform distribution. If $w \leq \exp(l_k(x*) - u_k(x*))$ then we accept x^*, if this condition is not verify x^* then we compute $h(x^*)$ and we accept the sample if $w \leq \exp(h(x^*) - u_k(x^*))$ otherwise we reject. We count the number of sample that we accept and stop when it reaches n.

3 Updating

After every new sample, if we evaluated $h(x^*)$ in the sampling step we need to update the variables. To do so we add x^* to T_k which is now T_{k+1} and sort T_{k+1}. Then we add the value $h(x^*)$ and $h'(x^*)$ to the vectors mentioned earlier and check that h is still concave. We can now do the same step as in the initialisation and compute a new s_k and z using the vector of T_{k+1}, $h(x_i)$ and $h'(x_i)$.

4 Testing

We decided to work on the following test:
-Tested that the ars function with several log concave density distribution: normal,logistic, gamma, laplace, uniform. We use the Kolmogorov–Smirnov test to compare the sample from the real density distribution and the one given by our algorithm.
-Tested that the function return an error if g is not a function or D_left, D_right, n are not numbers.
-Tested that the function return an error if the input g is not log-concave.
-Tested the case where either D is unbounded.

5 Team Work

Each of us was assigned a specific task in this group work. We assigned independents task to everyone so we could work on our own and then merge our individual work together. Jilin was in charge of coding the initialisation part and testing, Jason the sampling part and the PDF document and Tianchen the updating and testing.