Stat 243 Final Project Writeup

Vaibhav Ramamoorthy, Colin Kou, Brandon Mannion, and Jonathan Lee

12/7/2018

Functions

We created a very modular solution for our Adaptive-Rejection Sampler, which handled unique subtasks within individual functions. Below, we list each function we created and its modular purpose:

get_initial_abscissae()

This function builds the vector of initial abscissae. This function is called at the beginning of ars() to set the initial abscissae.

get_z() and get_z_all()

Together, these functions use the abscissae and the log density to find the vector of points at which the tangent lines at the abscissae intersect. The get_z_all() function is called within ars() to recreate the z vector each time the abscissae get updated.

get_u_segment() and get_u()

Together, these functions use the abscissae and the log density to build the piecewise upper bound of the log density using tangent lines at x. We represent this piecewise function as a list of slopes and intercepts rather than as a closure-type variable in order to more easily calculate integrals later on. The get_u() function is called within ars() to recreate the upper bound each time the abscissae get updated.

get_l_segment() and get_l()

These functions work very similarly to get_u_segment() and get_u(), but instead build the piecewise lower bound by calculating the slopes and intercepts of the chords between adjacent abscissae. Again, we represent this piecewise function as a list of slopes and intercepts. The get_l() function is called within ars() to recreate the lower bound each time the abscissae get updated.

get_s_integral()

This function calculates the integrals under each piecewise element of s, which is the normalized exponential of the u function. The function works by calculating the integral under each piecewise element of s analytically, using the formula as calculated below

$$\int_{z_1}^{z_2} exp(u_j(t))dt = \int_{z_1}^{z_2} exp(a_j+b_jt)dt = \int_{a_j+b_jz_1}^{a_j+b_jz_2} exp(y)\frac{1}{b_j}dy = \frac{1}{b_j}[exp(y)]_{a_j+b_jz_1}^{a_j+b_jz_2} = \frac{exp(a_j+b_jz_2) - exp(a_j+b_jz_1)}{b_j}$$

where a_j and b_j are the intercept and slope of the j^{th} segment of u respectively. The function returns a vector of integrals, of which the j^{th} element corresponds to the integral under the j^{th} segment of s.

sample.s()

This function performs the actual sampling. It samples n points from s by using the inverse transform sampling method as follows. We draw a vector q of length n from the Unif(0, 1) distribution. Then, for each element of q, we find an $x^\star$ such that $F_s(x^\star) = q$. To do so, we first have to identify the segment of u of which $x^\star$ is in the domain. Then, we calculate $x^\star$ by using the inverse CDF as follows:

$$\int_{z_j}^{x^\star} \frac{1}{I_s} exp(u_j(t))dt = q - F_s(z_j)$$

$$\int_{z_j}^{x^\star} exp(a_j + b_j t)dt = I_s(q - F_s(z_j))$$

$$\int_{a_j+b_j z_j}^{a_j+b_j x^\star} exp(y)\frac{1}{b_j}dy = I_s(q - F_s(z_j))$$

$$[exp(y)]_{a_j+b_j z_j}^{a_j+b_j x^\star} = b_j I_s(q - F_s(z_j))$$

$$exp(a_j + b_j x^\star) - exp(a_j + b_j z_j) = b_j I_s(q - F_s(z_j))$$

$$a_j + b_j x^\star = log(b_j I_s(q - F_s(z_j)) + exp(a_j + b_j z_j))$$

$$x^\star = \frac{log(b_j I_s(q - F_s(z_j)) + exp(a_j + b_j z_j)) - a_j}{b_j}$$

Using this formula, we can convert our q vector into a vector of $x^\star$. We return this vector.

Tests

Team Member Contributions