Stat 243 Final Project

Brian Collica, Ashish Ramesh, Ben Searchinger

Architecture

The code was written as a set of modular functions, each of which carries out a specific task. Each of these functions is then called by the main wrapper function `ars`.

Modular Functions

The functions we added include a function for generating tangent intersections for the given density from the abscissae provided, finding the upper hull from these tangent lines, sampling from the envelope created by the upper hull, and creating the lower hull from given abscissae. We included two additional helper functions: one for approximating a derivative via the finite difference method, and one to comprehensively check the inputs to the main wrapper function.

The function `tanIntersect()` calculates the tangent intersections to a given log-density, for tangent lines at a given set of abscissae. It does this by approximating the derivatives to the log-density at the given abscissae, and then using Equation (1) from Gilks et. al. to compute the x-coordinates of the intersections. This function returns a list containing the original abscissae, the x-coordinates of the interesections, and the values of the log-density and its derivatives at these intersections.

The function `upperHull()` corresponds mainly to Equation (2) from Gilks et. al., and calculates the y-values for the upper hull of the log-density at a given set of x-coordinates. To do this, it first calculates the tangent intersections for the segments forming the upper hull, using the `tanIntersect()` function. Then, it uses the values returned by this function and Equation (2) to compute the y-value on the hull corresponding to any x-value provided as an input. This is done in a segment-wise manner, exploiting the fact that the intersections returned by `tanIntersect()` correspond to the boundaries of each segment in order to partition the x-coordinates given based on which segment they fall into.

The function `sampleEnv()` corresponds most closely to Equation (3) from Gilks et. al., and samples `n` values from the upper hull for the log-density. It does this by first computing tangent intersections for a given log-density. Then, it finds the normalizing constant in Equation (3) by integrating numerically (using the `integrate()` function on `exp(upperHull(...))`). From here, samples are created using the inverse CDF method applied to each segment, using numerical integration to find the value of the CDF at each tangent intersection and using the explicit equation for the inverse of the CDF (derived below). The function includes a check to remove non-finite sample values and resample from the hull as needed.

[derive the inverse CDF here]

The function `lowerHull()` corresponds to Equation (4) of Gilks et. al., and computes y-values for the lower hull of the log-density. It does this by first calculating tangent intersections, and then uses Equation (4) to compute the appropriate y-values.

The function `approxD()` approximates the derivative of a given input function at a specified point, using a central finite-difference method. It is capable of returning first or second-order derivatives, and uses a value of `sqrt(.Machine$double.eps)` as its default finite difference, although it allows a user to specify a different difference value.

The function `checkThat()` serves as a comprehensive check of inputs to the main wrapper function. It includes checks for appropriate parameters, sample sizes, initial abscissae, and log-concavity of the density. It also generates starting values if none are provided.

Main `ars()` Function

Testing

We included tests for each individual modular function, as well as more comprehensive tests for the `ars()` function.

Tradeoffs

One major tradeoff in our approach is due in part to the sequential nature of the algorithm.

Individual Contributions

Brian wrote the `approxD()`, `tanInteresect()`, and `checkThat()` functions, as well as their tests. Ashish wrote the `upperHull()` and `sampleEnv()` functions, as well as tests for these and the `lowerHull()` function. Ben wrote the `lowerHull()` function and the main wrapper `ars()` function.