Adaptive Rejection Sampling

Matthew Seguin

Summary

Project Repository URL: GitHub

The goal of this project is to implement adaptive rejection sampling as described below on a differentiable, strictly log concave density where $\frac{dh}{dx} = \frac{d}{dx} \log f(x)$ is monotonically (strictly) decreasing.

Adaptive rejection sampling uses lower hull and upper hulls to squeeze a (possibly un-normalized) log concave density $f(x)$ defined in a connected domain D (i.e. with support over D) and sample from it according to the following:

- Find $h(x) = \log f(x)$ and $h'(x) = \frac{d}{dx} \log f(x)$
- Generate a k abscissae $T_k = \{x_j : j \in \{1, 2, ..., k\}\}$ in D such that $x_1 \leq x_2 \leq \cdots \leq x_k$. If D is unbounded on the left choose x_1 such that $h'(x_1) > 0$ and similarly if D is unbounded on the right choose x_k such that $h'(x_k) < 0$.
- Find the intersection points for the upper hull

  $$z_j = \frac{h(x_{j+1}) - h(x_j) - x_{j+1}h'(x_{j+1}) + x_j h'(x_j)}{h'(x_j) - h'(x_{j+1})} = \frac{(h(x_{j+1}) - x_{j+1}h'(x_{j+1})) - (h(x_j) - x_j h'(x_j))}{h'(x_j) - h'(x_{j+1})}$$

  The reason we rewrite it as the final expression is because it can be easily calculated in a vectorized manner in python.
- Find the upper hull $u_k(x)$ where $u_k(x) = h(x_j) + (x - x_j)h'(x_j)$ for $x \in [z_{j-1}, z_j]$ for each $j \in \{1, ..., k\}$ and $z_0 = \inf D$ (or $-\infty$ if D is unbounded on the left) and $z_k = \sup D$ (or ∞ if D is unbounded on the right)
- Find $s_k(x) = \frac{\exp u_k(x)}{\int_D \exp u_k(t)\, dt}$
- Find the lower hull $l_k(x)$ where $l_k(x) = \frac{(x_{j+1} - x)h(x_j) + (x - x_j)h(x_{j+1})}{x_{j+1} - x_j}$ for $x \in [x_j, x_{j+1}]$ for each $j \in \{1, ..., k-1\}$. Then for $x < x_1$ or $x > x_k$ we define $l_k(x) = -\infty$
- Sample x^* from $s_k(x)$ and independently sample $w \sim \text{Uniform}(0, 1)$
- If $w \leq \exp(l_k(x^*) - u_k(x^*))$ then accept x^* and continue from the step where we sample from $s_k(x)$ if we don't already have n points, otherwise continue to the following step.
- Evaluate $h(x^*)$ and $h'(x^*)$. Then if $w \leq \exp(h(x^*) - u_k(x^*))$ accept x^*, otherwise we reject x^*.
- If $h(x^*)$ and $h'(x^*)$ were evaluated in the sampling process add x^* to T_k to form T_{k+1} and repeat the steps starting from after we found the abscissae until n points have been sampled.

Process of Implementation

The first thing to notice is that we will be adding points over time to our abscissae so we should have separate functions outside of the main ars function used to calculate the new intersection points, hulls, and envelope.

Sampling x^* from $s_k(x)$

The next thing to note is that we need to sample x^* from $s_k(x)$ so we need to develop a methodology of doing that. Such a method is described below:

First note that $s_k(x) = \frac{\exp u_k(x)}{\int_D \exp u_k(t)\,dt}$ is a piecewise function. So we can first sample a categorical variable that tells us which range to choose from. Then based on that we can sample from inside that range.

- Define range j to be $R_j = [z_{j-1}, z_j]$ for each $j \in \{1, ..., k\}$. Then

$$\mathbb{P}[x^* \in R_j] = \int_{R_j} \frac{\exp u_k(x)}{\int_D \exp u_k(t)\,dt}\,dx$$

This looks like a lot but it is easy to break down. First note that $\bigcup_j R_j = D$. Now we will evaluate simpler integrals.

$$\int_{R_j} \exp u_k(x)\,dx = \int_{z_{j-1}}^{z_j} \exp(h(x_j) + (x - x_j)h'(x_j))\,dx = \exp(h(x_j) - x_j h'(x_j)) \int_{z_{j-1}}^{z_j} \exp(xh'(x_j))\,dx$$

Now there are two cases $h'(x_j) = 0$ vs $h'(x_j) \neq 0$.
First if $h'(x_j) = 0$:

$$\int_{R_j} \exp u_k(x)\,dx = \exp(h(x_j) - x_j h'(x_j)) \int_{z_{j-1}}^{z_j} \exp(xh'(x_j))\,dx = \exp h(x_j) \int_{z_{j-1}}^{z_j} 1\,dx = (z_j - z_{j-1}) \exp h(x_j)$$

Now if $h'(x_j) \neq 0$

$$\int_{R_j} \exp u_k(x)dx = \exp(h(x_j) - x_j h'(x_j)) \int_{z_{j-1}}^{z_j} \exp(xh'(x_j))dx = \exp(h(x_j) - x_j h'(x_j)) \left(\frac{1}{h'(x_j)} \exp(xh'(x_j)) \Big|_{z_{j-1}}^{z_j} \right)$$

$$= \exp(h(x_j) - x_j h'(x_j)) \left(\frac{\exp(z_j h'(x_j)) - \exp(z_{j-1} h'(x_j))}{h'(x_j)} \right)$$

Quickly note that $\int_D \exp u_k(t)\,dt = \sum_{j=1}^k \int_{R_j} \exp u_k(x)\,dx$
Therefore if we let $w_j = \int_{R_j} \exp u_k(x)\,dx$ we see that:

$$\mathbb{P}[x^* \in R_j] = \int_{R_j} \frac{\exp u_k(x)}{\int_D \exp u_k(t)\,dt}\,dx = \frac{1}{\int_D \exp u_k(t)\,dt} \int_{R_j} \exp u_k(x)\,dx$$

$$= \frac{1}{\sum_{m=1}^k \int_{R_m} \exp u_k(x)\,dx} \int_{R_j} \exp u_k(x)\,dx = \frac{w_j}{\sum_{m=1}^k w_m} = p_j$$

Now we know how to sample from a probability vector like this, there are k possible outcomes we need to choose one. So we have successfully sampled from the ranges.

- Now assuming we are inside of R_j we need to sample a value from in that range. The denisty is easily seen to be $\frac{\exp(h(x_j)+(x-x_j)h'(x_j))}{w_j}$ since we are just normalizing for already being in R_j.
  Again we have two cases $h'(x_j) = 0$ vs $h'(x_j) \neq 0$.

  First if $h'(x_j) = 0$:
  If $h'(x_j) = 0$ this is clearly a uniform random variable over $[z_{j-1}, z_j]$ as the density does not depend on x so we can just sample a uniform.

  Now if $h'(x_j) \neq 0$:
  If $h'(x_j) \neq 0$ we get the following CDF for $x^*|x^* \in R_j$:

$$F(x) = \int_{z_{j-1}}^{x} \frac{\exp(h(x_j) + (x - x_j)h'(x_j))}{w_j}\, dx = \frac{\exp(h(x_j) - x_j h'(x_j))}{w_j} \int_{z_{j-1}}^{x} \exp(xh'(x_j))\, dx$$

$$= \frac{\exp(h(x_j) - x_j h'(x_j))}{w_j} \left(\frac{1}{h'(x_j)} \exp(xh'(x_j)) \Big|_{z_{j-1}}^{x} \right)$$

$$= \frac{1}{w_j} \left(\frac{\exp(h(x_j) - x_j h'(x_j))}{h'(x_j)} \right) \left(\exp(xh'(x_j)) - \exp(z_{j-1}h'(x_j)) \right)$$

  Quickly note that

$$\frac{1}{w_j} = \left(\frac{1}{\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j))} \right) \left(\frac{h'(x_j)}{\exp(h(x_j) - x_j h'(x_j))} \right)$$

  Continuing we get

$$F(x) = \frac{1}{w_j} = \left(\frac{1}{\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j))} \right) \left(\frac{h'(x_j)}{\exp(h(x_j) - x_j h'(x_j))} \right)$$

$$= \left(\frac{1}{\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j))} \right) \left(\exp(xh'(x_j)) - \exp(z_{j-1}h'(x_j)) \right) = \frac{\exp(xh'(x_j)) - \exp(z_{j-1}h'(x_j))}{\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j))}$$

  Since the term in the w_j cancels out to simplify to the above. We can then use the inverse CDF method to sample from this. Namely sample $u \sim \text{Uniform}(0, 1)$ then our sample for x^* is $F^{-1}(u)$. This is solved below:
  If $F^{-1}(u) = x^*$ then $u = F(x^*)$ so

$$u = F(x^*) = \frac{\exp(x^* h'(x_j)) - \exp(z_{j-1}h'(x_j))}{\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j))}$$

$$u\Big(\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j)) \Big) = \exp(x^* h'(x_j)) - \exp(z_{j-1}h'(x_j))$$

$$u\Big(\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j)) \Big) + \exp(z_{j-1}h'(x_j)) = \exp(x^* h'(x_j))$$

$$\log\left(u\Big(\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j)) \Big) + \exp(z_{j-1}h'(x_j)) \right) = x^* h'(x_j)$$

$$x^* = \frac{\log\left(u\Big(\exp(z_j h'(x_j)) - \exp(z_{j-1}h'(x_j)) \Big) + \exp(z_{j-1}h'(x_j)) \right)}{h'(x_j)}$$

$$= \frac{\log\left(u \exp(z_j h'(x_j)) + (1 - u) \exp(z_{j-1}h'(x_j)) \right)}{h'(x_j)}$$

Quickly note that the cases of $z_0 = -\infty$ and $z_k = \infty$ are not an issue for these integrals converging or the probabilities being positive. Since in these cases we chose x_1 such that $h'(x_1) > 0$ and similarly x_k such that $h'(x_k) < 0$. This implies that $z_0 h'(x_1) = -\infty$ and $z_k h'(x_k) = -\infty$ which tells us:

$$w_1 = \exp(h(x_1) - x_1 h'(x_1)) \left(\frac{\exp(z_1 h'(x_1)) - \exp(z_0 h'(x_1))}{h'(x_1)} \right)$$

$$= \frac{\exp(h(x_1) - x_1 h'(x_1))}{h'(x_1)} \left(\exp(z_1 h'(x_1)) - \exp(z_0 h'(x_1)) \right) = \frac{\exp(h(x_1) - x_1 h'(x_1))}{h'(x_1)} \left(\exp(z_1 h'(x_1)) - \exp(-\infty) \right)$$

$$= \frac{\exp(h(x_1) - x_1 h'(x_1))}{h'(x_1)} \left(\exp(z_1 h'(x_1)) \right) > 0$$

Because $h'(x_1) > 0$. Similarly:

$$w_k = \exp(h(x_k) - x_k h'(x_k)) \left(\frac{\exp(z_k h'(x_k)) - \exp(z_{k-1} h'(x_k))}{h'(x_k)} \right)$$

$$= \frac{\exp(h(x_k) - x_k h'(x_k))}{h'(x_k)} \left(\exp(z_k h'(x_k)) - \exp(z_{k-1} h'(x_k)) \right)$$

$$= \frac{\exp(h(x_k) - x_k h'(x_k))}{h'(x_k)} \left(\exp(-\infty) - \exp(z_{k-1} h'(x_k)) \right) = \frac{\exp(h(x_k) - x_k h'(x_k))}{h'(x_k)} \left(-\exp(z_{k-1} h'(x_k)) \right) > 0$$

Because $h'(x_k) < 0$

Generating the initial abscissae T_k

The next thing to note is that generating the initial abscissae is not trivial based on the provided domain:

We firstly note that since the (possibly un-normalized) density is log concave it is also unimodal. Therefore there exists some x_0 such that $f(x)$ is monotonically increasing for $x < x_0$ and is monotonically decreasing for $x > x_0$.

We want to generate our upper and lower hulls in order to encompass a decent part of the density, which in this case means generating them around the mode. So we want to generate our abscissae around the mode.

If the domain is unbounded on either side there are infinitely many places the mode could be. This introduces a need to implement some sort of maximization algorithm for finding the mode (maximum of the function). However, in any of these algorithms a point need always be supplied. Without any prior knowledge about the density function the point might be so far away that any algorithm would never converge. Note that if the domain is unbounded we can not simply start at a point and increment it until we find the function is non-zero because we might skip over the whole distribution (even with increments decreasing in size). Therefore one of the inputs for our function is a starting point for the maximization.

Since the function is unimodal (and by assumption the derivative exists) we know the derivative will be positive up until the mode (where it equals 0) then will be negative so finding the mode amounts to finding the zero of the derivative.

Another property that is useful is that $h'(x_0) = \frac{d}{dx} h(x)\Big|_{x=x_0} = \frac{d}{dx} \log f(x)\Big|_{x=x_0} = \frac{f'(x)}{f(x)}\Big|_{x=x_0} = \frac{f'(x_0)}{f(x_0)} = 0$ when x_0 is the mode. This is useful because the rate of change of the density might be very small at some points so if we try to directly apply an algorithm on $f'(x)$ it might think we started at the root so working on the log scale could be important.

After we have the mode we can just take k linearly spaced points centered around it for our abscissae T_k.

Functions

The main function ars:

- Inputs: the (possibly un-normalized) density $f(x)$ (labeled as f), the support (or domain) D (labeled as support), the number of points to sample n (labeled as n), a starting point x_0 (labeled as x0) close to the mode of the function which is defaulted to None in case the domain is bounded, the number of points in the initialization step k (labeled as k) which is defaulted to 10
- Purpose: uses other functions to get parameters and implement adaptive rejection sampling.
- Output(s): the sampled points

The abscissae generating function:

- Inputs: the derivative of the log of $f(x)$, $h'(x)$ (labeled as dh), the support (or domain) D (labeled as support), a starting point x_0 (labeled as x0) close to the mode of the function which is defaulted to None in case the domain is bounded, the number of points to generate k (labeled as k)
- Purpose: uses scipy to solve $h'(x) = 0$ (the mode of $f(x)$) then takes linearly spaced points around the mode whose dh values are not too small in absolute value.
- Output(s): the initial abscissae T_k

The upper hull intercept finding function:

- Inputs: the abscissae points $\{x_1, ..., x_k\}$ (labeled as x_vals), the log density evaluated at these points $\{h(x_1), ..., h(x_k)\}$ (labeled as h_vals), the derivative of the log density evaluated at these points $\{h'(x_1), ..., h'(x_k)\}$ (labeled as dh_vals), the support (or domain) D (labeled as support)
- Purpose: finds the intersection points of the upper hull and returns the points as well as the heights of the tangent lines of $h(x)$ from the abscissae at $x = 0$ which are used later.
- Output(s): the intersection points of the upper hull and the heights of the tangent lines of $h(x)$ from the abscissae at $x = 0$

The upper hull function:

- Inputs: the intersection points $\{z_0, z_1, ..., z_{k-1}, z_k\}$ (labeled as z), the abscissae points $\{x_1, ..., x_k\}$ (labeled as x_vals), the log density evaluated at these points $\{h(x_1), ..., h(x_k)\}$ (labeled as h_vals), the derivative of the log density evaluated at these points $\{h'(x_1), ..., h'(x_k)\}$ (labeled as dh_vals), the point(s) to be evaluated at x (labeled as x)
- Purpose: finds which range x lies in and returns the (unexponentiated) upper hull function at x.
- Output(s): the (unexponentiated) upper hull function at x

The lower hull function:

- Inputs: the abscissae points $\{x_1, ..., x_k\}$ (labeled as x_vals), the log density evaluated at these points $\{h(x_1), ..., h(x_k)\}$ (labeled as h_vals), the point(s) to be evaluated at x (labeled as x)
- Purpose: finds which range x lies in and returns the (unexponentiated) lower hull function at x.
- Output(s): the (unexponentiated) lower hull function at x.

The range probability finder function:

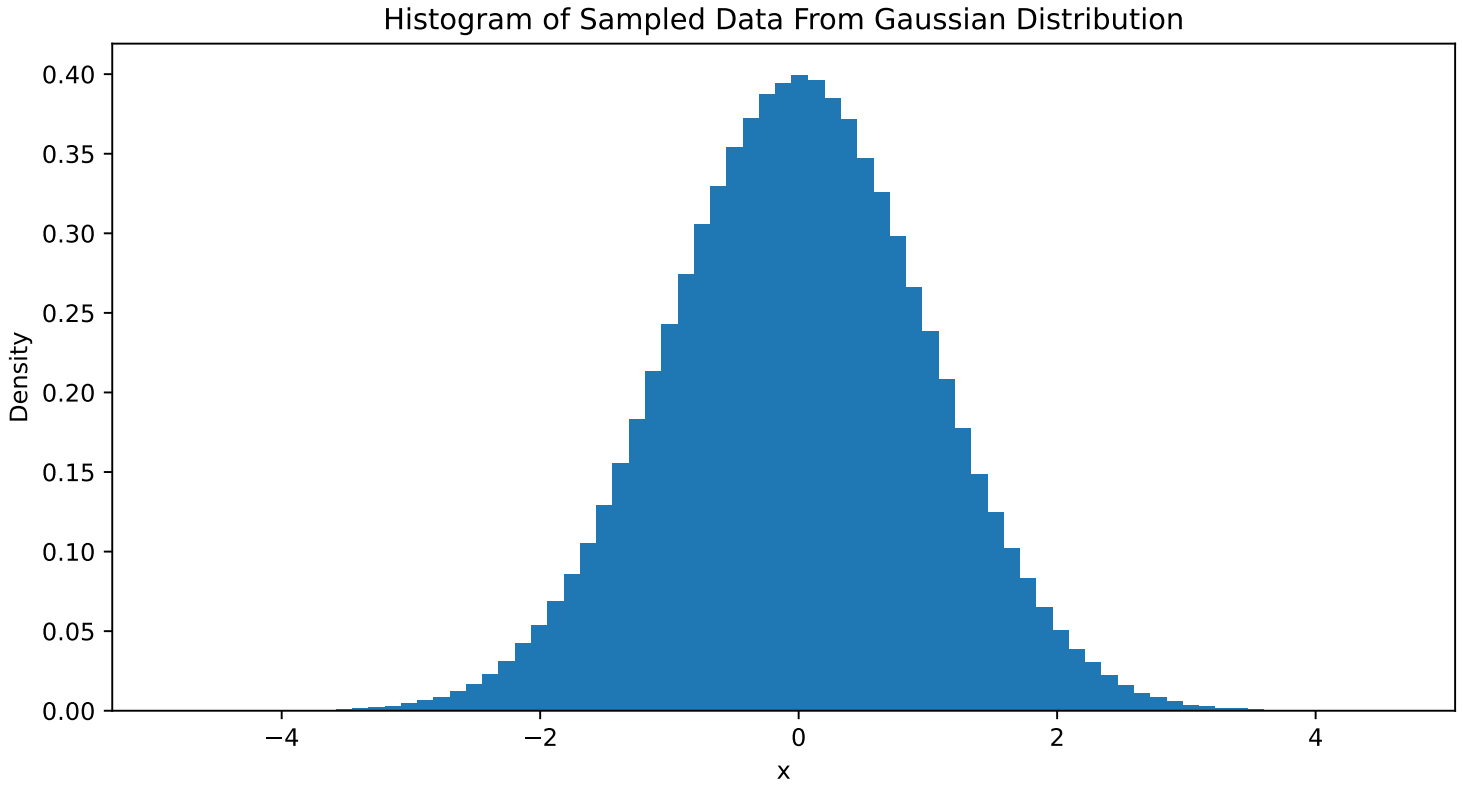
- Inputs: the upper hull intersection points $\{z_0, z_1, ..., z_{k-1}, z_k\}$ (labeled as z), the heights of the tangent lines of $h(x)$ at $x = 0$ (labeled as heights), the derivative of the log density evaluated at these points $\{h'(x_1), ..., h'(x_k)\}$ (labeled as dh_vals)
- Purpose: finds the probability of being in each range according to s_k and returns those probabilities.
- Output(s): the probabilities of being in each range according to s_k

The $s_k(x)$ sampler function:

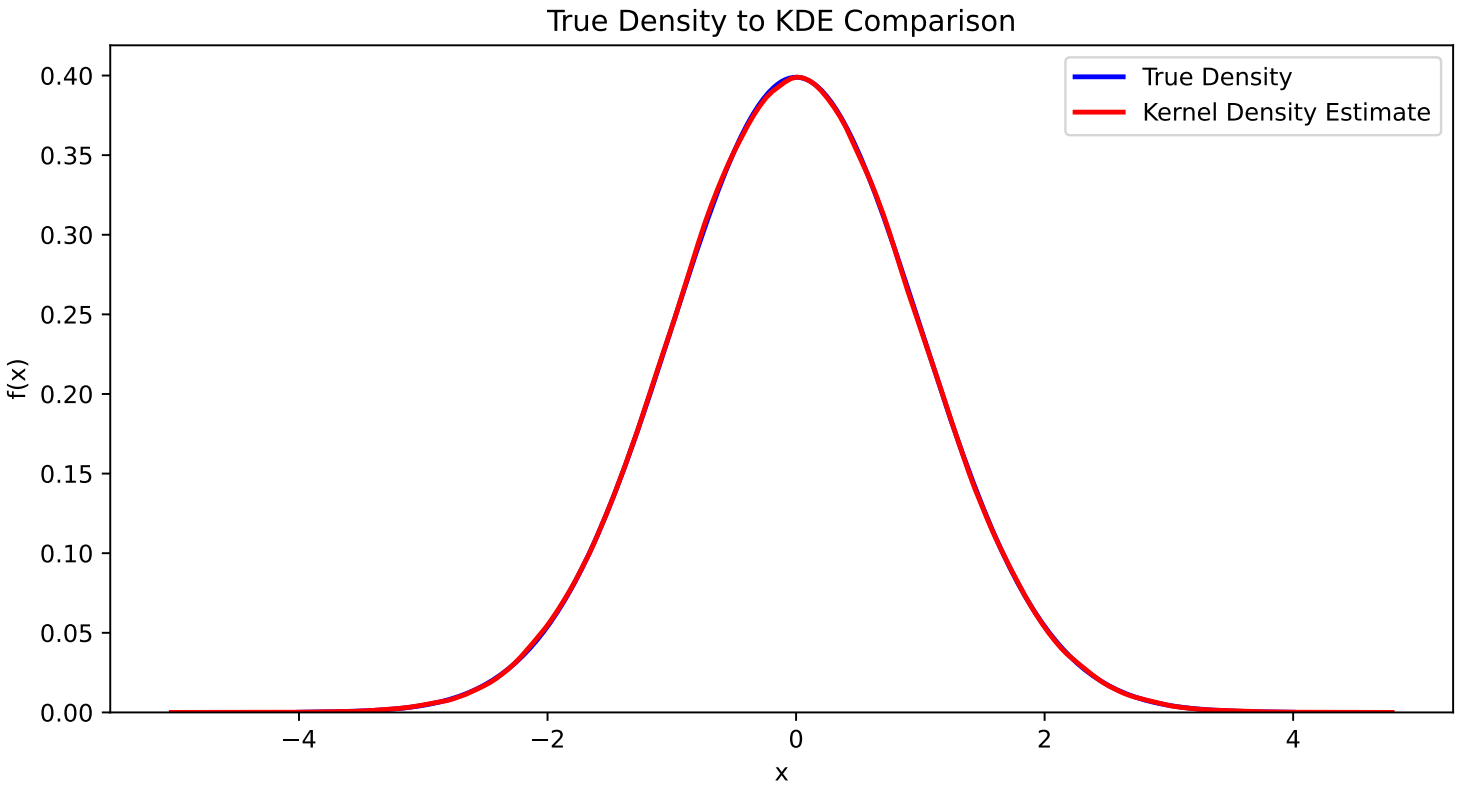
- Inputs: the probabilities of being in each range according to s_k (labeled as probs), the upper hull intersection points $\{z_0, z_1, ..., z_{k-1}, z_k\}$ (labeled as z), the derivative of the log density evaluated at these points $\{h'(x_1), ..., h'(x_k)\}$ (labeled as dh_vals), the number of points to sample (labeled as num)
- Purpose: sample from $s_k(x) = \frac{\exp u_k(x)}{\int_D \exp u_k(t)\, dt}$.
- Output(s): the sampled value(s) from s_k

Testing Performance

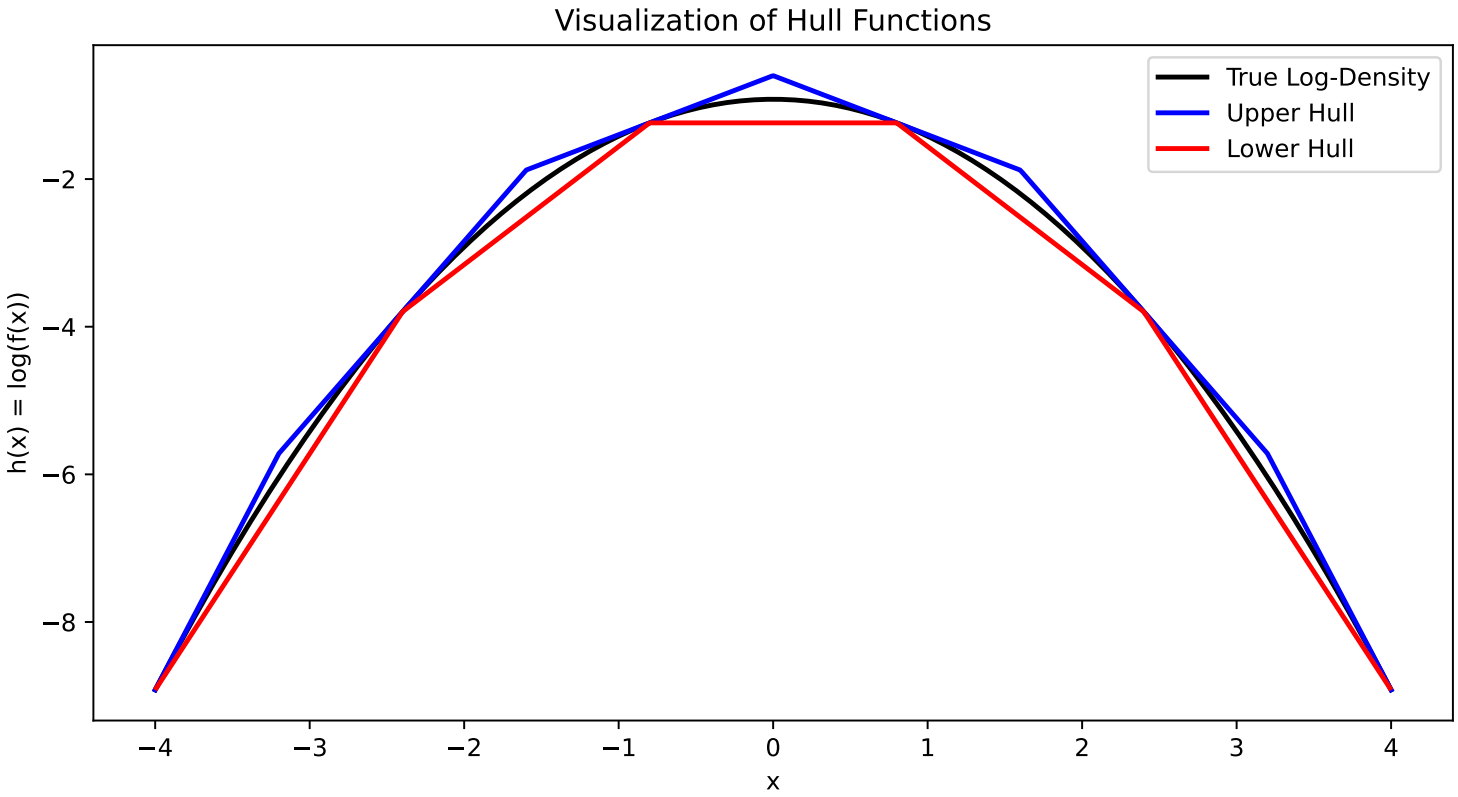
```
from scipy.stats import norm as gaussian
f = lambda x: gaussian.pdf(x)
from ars import *
sampler = AdaptiveRejectionSampler(f, [-np.inf, np.inf])
_ = sampler.ars(1000000, 5.0)
```



The sample clearly looks normal. Now comparing the kernel density estimate of our data to the true density.



The two are nearly identical. Now taking a look at how the upper and lower hulls look like around the true log-density.



Areas for improvement

- Initial abscissae generation on infinite domains is not very robust.
- Possible gains by implementing caching for calls on h and h', but performance could worsen considering the simplicity of such function calls.
- Further analysis to determine the optimal number of points to sample at once. The algorithm currently samples $\text{floor}(n/10)$ points at each iteration.
- Expanding compatibility with other common packages.