AdaptiveIS Package

Sheng Xu, 4302 41 552

14 November, 2016

Usage

The *ais* function approximates the expected value of a function of iid standard uniform random variables via adaptive importance sampling, where the importance sampling parameter is updated using robust stochastic approximation.

```
ais(f::Function,d::Int64;n::Int64=10^4,t0=zeros(d),lb=t0-0.5,
        ub=t0+0.5,npart::Int64=5,sampsize::Int64=10^2,
        accel::AbstractString="none",dimreduc::Bool=false)
```

f a function of iid standard uniform random variables.

d the dimension of the domain of f.

n the number of iterations of the Monte Carlo simulation.

t0 a vector of 1's and 0's of length d which specifies the importance sampling distribution. Each 1 sets the importance sampling distribution to be the exponential distribution in the corresponding dimension, and each 0 the normal distribution with unit standard deviation.

lb a vector of length d which denotes the lower bound of the domain of the importance sampling parameter in each dimension.

ub a vector of length d which denotes the upper bound of the domain of the importance sampling parameter in each dimension.

npart the number of points to discretise each dimension of the domain of the importance sampling parameter into. Used in calculating the step size in robust stochastic approximation and choosing the auxiliary parameter via sample average approximation.

sampsize the number of samples to generate at each point in the discretised domain of the importance sampling parameter when calculating the step size in robust stochastic approximation.

accel specifies whether an auxiliary parameter is used to accelerate the Monte Carlo simulation, and if so, the method of choosing the auxiliary parameter. Useful when the function f is zero with a high probability. Accepted arguments are "none", "directsub", "sa", and "saa".

dimreduc if true, implements dimension reduction by restricting all components of the importance sampling parameter to be equal. Useful when d is large (e.g. at least 5) and f is symmetric with respect to its inputs (so that the restriction is reasonable).

1 Introduction

The goal is to use Monte Carlo simulations to estimate some constant C which can be expressed as

$$C \coloneqq \mathbb{E}[f(U)],$$

where $U \sim U(0,1)^d$ is uniformly distributed on the unit d-dimensional hypercube $(0,1)^d$ and f is a function from $(0,1)^d$ to $\mathbb{R}$. Alternatively, U can be thought of as a vector in $\mathbb{R}^d$ composed of d iid standard uniform random variables. It is recommended that the output of f be of the type **Float64**. Let n denote the total number of iterations of the Monte Carlo simulation, which is assumed to be fixed a priori. Rather than using the crude estimator

$$\frac{1}{n}\sum_{k=1}^{n} f(U_k) \approx C,$$

where U_k are iid with the same distribution as U, an estimator with (potentially) lower variance is used – the adaptive importance sampler.

2 Importance Sampling

Let ϕ denote the pdf of the standard normal distribution. In this package, the importance sampling density is assumed to be the joint pdf of independent exponential and/or normal random variables with unit standard deviation. That is,

$$g(z;\theta) = \prod_{k=1}^{d} g_k(z_k;\theta_k), \text{ where } g_k(z_k;\theta_k) = \theta_k e^{-\theta_k z_k} \text{ or } g_k(z_k;\theta_k) = \phi(z_k - \theta_k).$$

Let θ_0 (*t0* in Julia) be the d-dimensional vector of 'natural' parameters for $g(z;\theta)$, that is, 0 for each normal component and 1 for each exponential component. Specifying *t0* allows the user to choose the importance sampling density $g(z;\theta)$ – whether each component should be exponential or normal. Let $\Theta_0 \subseteq \mathbb{R}^d$ denote the natural domain of θ – the real line for each normal component and the positive half line for each exponential component. Finally, let $G(z;\theta)$ denote the corresponding cdf of $g(z;\theta)$.

For any $\theta \in \Theta_0$, the importance sampling estimator for C can be derived as follows:

$$C = \int_{(0,1)^d} f(u)\, du = \int_{(0,1)^d} f(G(G^{-1}(u;\theta);\theta_0)) \frac{g(G^{-1}(u;\theta);\theta_0)}{g(G^{-1}(u;\theta);\theta)}\, du.$$

The variance of this importance sampling estimator is given by

$$\begin{aligned}\int_{(0,1)^d} \left(f(G(G^{-1}(u;\theta);\theta_0)) \frac{g(G^{-1}(u;\theta);\theta_0)}{g(G^{-1}(u;\theta);\theta)} \right)^2 du - C^2 &= \int_{(0,1)^d} |f(u)|^2 \frac{g(G^{-1}(u;\theta_0);\theta_0)}{g(G^{-1}(u;\theta_0);\theta)}\, du - C^2 \\ &=: V(\theta) - C^2.\end{aligned}$$

Define the functions

$$H(u;\theta,\lambda) \coloneqq \frac{g(G^{-1}(u;\lambda);\theta_0)}{g(G^{-1}(u;\lambda);\theta)}, \quad R(u;\theta) \coloneqq f(G(G^{-1}(u;\theta);\theta_0))H(u;\theta,\theta),$$

so that

$$C = \mathbb{E}[R(U;\theta)], \quad V(\theta) = \mathbb{E}[|f(U)|^2 H(U;\theta,\theta_0)], \quad \nabla_\theta V(\theta) = \mathbb{E}[|f(U)|^2 \nabla_\theta H(U;\theta,\theta_0)].$$

Note that, if $\theta = \theta_0$, then this importance sampler reduces to the crude estimator, that is, $R(U;\theta_0) = f(U)$. The interchange of the expectation and derivative above holds under certain regularity conditions which will not be discussed here. θ is chosen adaptively, that is, the search for θ^*, the minimiser of V, is run simultaneously with the Monte Carlo simulation for C.

3 Robust Stochastic Approximation

This package uses robust stochastic approximation to adaptively search for θ^*. First, it is assumed that θ is restricted to $\Theta \subseteq \Theta_0$, a Cartesian product of d closed intervals which contains θ_0. In Julia, the lower endpoints of the intervals correspond to the d-dimensional vector *lb* and the upper endpoints *ub*. Introduce the following notation:

$$D_\Theta(\xi) := \max_{\theta\in\Theta} |\theta - \xi|, \quad L_\Theta := \max_{\theta\in\Theta} \mathbb{E}[|f(U)|^4 |\nabla_\theta H(U;\theta,\theta_0)|^2]^{\frac{1}{2}}.$$

Let $\Pi_\Theta\{\cdot\}$ denote the metric projection onto Θ. The robust algorithm is given by

$$\begin{cases} \bar{\theta}_0^k = \dfrac{1}{k}\displaystyle\sum_{l=0}^{k-1} \theta_l \\ \text{generate } R(U_k;\bar{\theta}_0^k) \\ \theta_k = \Pi_\Theta\{\theta_{k-1} - \gamma |f(U_k)|^2 \nabla_\theta H(U_k;\theta_{k-1},\theta_0)\} \end{cases} \qquad k = 1,\dots,n, \tag{1}$$

where

$$\gamma = \frac{D_\Theta(\theta_0)}{L_\Theta\sqrt{n}} \sqrt{\sum_{k=1}^{n} \frac{1}{k}}.$$

It is assumed that the user does not know a priori what a good choice of θ is. Under this assumption, θ is initialised at θ_0, the vector of natural parameters, as this recovers the crude estimator. To compute L_Θ, the space Θ is discretised – each dimension is discretised into *npart* equally spaced points so that Θ is discretised into $npart^d$ points, and the expectation is approximated with the mean of *sampsize* samples at each point in the discretised space. The number of calculations to compute L_Θ grows exponentially with respect to d, and is the limiting factor in terms of speed for the *ais* function.

The *ais* function returns an object of the type **Ais**. Calling an object of this type displays

$$\mu_n := \frac{1}{n}\sum_{k=1}^{n} R(U_k;\bar{\theta}_0^k) \approx C,$$

which is the empirical mean after n iterations, and $\bar{\theta}_0^n$, the estimate of θ^* after n iterations. The intermediate values μ_k and $\bar{\theta}_0^k$ for $k = 1,\dots,n$ are stored in the fields μ and θ, respectively. They can be plotted to check for convergence. Plotting an object of the type *Ais* yields a plot of its field μ.

4 Acceleration

Algorithm (1) can be slow when the probability $\mathbb{P}(f(U) = 0)$ is high, as the event $\{f(U_k) = 0\}$ implies $\theta_k = \theta_{k-1}$. If it is known a priori that $\mathbb{P}(f(U) = 0)$ is high, say, larger than 99%, then this problem can be mitigated by introducing an auxiliary parameter λ. Introduce $\lambda \in \Theta_0$ to V as follows:

$$V(\theta) = \int_{(0,1)^d} |f(G(G^{-1}(u;\lambda);\theta_0))|^2 H(u;\theta,\lambda) H(u;\lambda,\lambda)\, du =: \mathbb{E}[N(U;\theta,\lambda)].$$

Again, note that if $\lambda = \theta_0$, then this reduces to the importance sampling framework in Section 2. Under the same regularity conditions as before, the expectation and derivative can be interchanged:

$$\nabla V(\theta) = \mathbb{E}[\nabla_\theta N(U;\theta,\lambda)].$$

For certain values of λ, this new estimator of ∇V is zero less often than the original estimator, $|f(U)|^2\nabla_\theta H(U;\theta,\theta_0)$. The definition of L_Θ is now generalised as a function of λ:

$$L_\Theta(\lambda) := \max_{\theta\in\Theta} \mathbb{E}[|\nabla_\theta N(U;\theta,\lambda)|^2]^{\frac{1}{2}}.$$

For any fixed λ, the robust algorithm is given by

$$\begin{cases} \bar{\theta}_0^k = \dfrac{1}{k}\displaystyle\sum_{l=0}^{k-1}\theta_l \\ \text{generate } R(U_k;\bar{\theta}_0^k) \\ \theta_k = \Pi_\Theta\{\theta_{k-1} - \gamma\nabla_\theta N(U_k;\theta_{k-1},\lambda)\} \end{cases} \qquad k = 1,\ldots,n, \tag{2}$$

where

$$\gamma = \frac{D_\Theta(\theta_0)}{L_\Theta(\lambda)\sqrt{n}}\sqrt{\sum_{k=1}^{n}\frac{1}{k}}.$$

We define $\tau := \min\{k \mid f(U_k) \neq 0\}$ and $\tau_n := \tau \wedge n$. To choose λ, Algorithm (1) is run for τ_n iterations, that is, until the first time θ updates. Then these τ_n iterations are used to choose λ, which is used to run Algorithm (2) for $n - \tau_n$ iterations.

This package contains three methods of choosing λ based on the first τ_n iterations. Regardless of the method, the value of λ is stored in the field λ.

4.1 accel="directsub"

Notice that $\nabla_\theta N(u;\theta,\lambda)$ contains $R(u;\lambda)$. Hence, good values of θ might also be good for λ, so this method chooses λ to be $\lambda = \bar{\theta}_0^{\tau_n+1}$. It is computationally very easy, but typically it does not accelerate the simulation by much.

4.2 accel="sa"

For any fixed λ, the probability that Algorithm (2) updates θ is given by

$$W(\lambda) := \mathbb{P}(f(G(G^{-1}(U;\lambda);\theta_0)) \neq 0).$$

This method chooses λ by using stochastic approximation to maximise W. It can be shown that

$$\nabla_\lambda W(\lambda) = \mathbb{E}[\mathbb{1}\{f(G(G^{-1}(U;\lambda);\theta_0)) \neq 0\}(T(G^{-1}(U;\lambda)) + \nabla_\lambda Q(\lambda))],$$

where $T(z) = z$ and $\nabla_\lambda Q(\lambda) = -\lambda$ for each normal component of $g(z;\theta)$, and $T(z) = -z$ and $\nabla_\lambda Q(\lambda) = 1/\lambda$ for each exponential component. λ is then chosen via

$$\lambda = \Pi_\Theta\{\theta_0 + (T(G^{-1}(U_{\tau_n};\theta_0)) + \nabla_\lambda Q(\theta_0))/\tau_n^{0.7}\}.$$

This method is also computationally very easy, but typically it does not accelerate the simulation by much.

4.3 accel="saa"

W can be rewritten as

$$W(\lambda) = \mathbb{E}\left[\mathbb{1}\{f(U) \neq 0\}\frac{1}{H(U;\lambda,\theta_0)}\right].$$

Instead of maximising this deterministic function, we can instead maximise its sample average approximation, that is,

$$\lambda = \underset{\lambda' \in \Theta}{\arg\min}\{H(U_{\tau_n};\lambda',\theta_0)\}.$$

Again, this computation is done by discretising the space Θ into $npart^d$ points. Compared to the other two methods, this method is computationally intensive, but typically yields better results.

5 Examples

5.1 One-Dimensional Example

The price of a European call option in the Black-Scholes model is given by

$$C = \mathbb{E}\left[e^{-rT}\max\left(S_0\exp\left[\left(r-\frac{\sigma^2}{2}\right)T+\sigma\sqrt{T}\Phi^{-1}(U)\right]-K,0\right)\right] =: \mathbb{E}[f(U)],$$

where Φ denotes the cdf of the standard normal distribution. We set $r = 0.05$, $T = 0.25$, $S_0 = 31$, $\sigma = 0.1$, and $K = 35$.

```
using AdaptiveIS
f(x)=exp(-0.05*0.25)*maximum([31.*exp((0.05-0.1^2/2.)*
        0.25+0.1*sqrt(0.25)*quantile(Normal(),x[1]))-35.;0.])
sim=ais(f,1)
plot(sim)
```

5.2 Three-Dimensional Example

Suppose we want to estimate the probability

$$C = \mathbb{P}\left(\frac{1}{3}(U^{(1)}+U^{(2)}+U^{(3)}) \geq 0.85\right) = \mathbb{E}\left[\mathbb{1}\left\{\frac{1}{3}(U^{(1)}+U^{(2)}+U^{(3)}) \geq 0.85\right\}\right] =: \mathbb{E}[f(U)].$$

```
using AdaptiveIS
f(x)=mean(x)>=0.85 ? 1. : 0.
sim=ais(f,3,t0=[1;1;0],accel="saa")
plot(sim.θ)
```