diagis: Diagnostic plot and multivariate summary statistics of weighted samples from importance sampling

Jouni Helske
October 27, 2016

Introduction

diagis is an R package containing functions relating weighted samples obtained for example from importance sampling. The main motivation for developing diagis was to enable easy computation of summary statistics and diagnostics of the weighted MCMC runs provided by bssm package (Helske and Vihola 2016; Vihola, Helske, and Franks 2017) for Bayesian state space modelling. For more broader use, the diagis package provides functions for computing weighted means and covariances of possibly multivariate samples, the running versions of these, as well as diagnostic plot function weight_plot for graphical diagnostic of weights.

All the mean and covariance functions are written in C++ using Rcpp (Eddelbuettel and François 2011; Eddelbuettel 2013) and RcppArmadillo (Eddelbuettel and Sanderson 2014) packages, making these function computationally very efficient even for large samples. The weight diagnostic plot uses ggplot (Wickham 2009) for visually appealing graphics, while gridExtra (Auguie 2016) combines the plots together.

Illustrations

As an illustration, consider estimating the expected value of Gamma($\alpha, \beta$) distribution using importance sampling. The density of Gamma($\alpha, \beta$) is

\[ p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x). \]

We will use another Gamma distribution with parameters $a$ and $b$ as our proposal distribution $q(x)$, so the weights are of form

\[ w(x) = \frac{p(x)}{q(x)} = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x) \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx) = \frac{\Gamma(a)\beta^\alpha}{\Gamma(\alpha)b^a} x^{a-\alpha} \exp(-(\beta-b)x), \]

and our importance sampling estimator for $\theta = \mathbb{E}_p(X)$ is

\[ \hat{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} Y_i w(Y_i), \]

where $Y_i, i = 1, \ldots, n$ are drawn from Gamma($a, b$). However, in practice we often have access only to unnormalized weights $w_u(x) = cw(x)$, which leads to self-normalized importance sampling estimate

\[ \tilde{\theta}_n = \frac{1}{\sum_{i=1}^{n} w_u(Y_i)} \sum_{i=1}^{n} Y_i w_u(Y_i). \]

Note that the self-normalized estimate $\tilde{\theta}_n$ is not unbiased, but still consistent. Also, we might want to use self-normalized version even when the computation of weights $w$ is possible (Owen 2013). Thus the functions in diagis are focused on self-normalized importance sampling as it is more generally applicable approach.
We can in principle choose $a$ and $b$ arbitrarily, but weights $w(x)$ have finite variance only if $a < \alpha$ and $b < \beta$. As we will soon see, infinite variance of the weights can make the importance sampling unreliable.

Let us first take $\alpha = 2$, and $\beta = 1$. The expected value is then $\alpha/\beta = 2$. For proposal distribution, let us first use $a = 1$ and $b = 0.5$:

```r
library("diagis")
set.seed(1)
x <- rgamma(10000, 1, 0.75)
w <- dgamma(x, 2, 1) / dgamma(x, 1, 0.75)
plot(w)
```

```
## weighted_mean(x, w)

# [1] 2.012761
```

Our estimate is fairly close to the theoretical value, and the plot of weights $w$ suggest that the distribution of the weights is nearly uniform with upper bound slightly below 2 (actual bound can be computed theoretically based on $w(x)$).

Let us now change $a$ to 2:

```r
set.seed(1)
x_bad <- rgamma(10000, 1, 2)
w_bad <- dgamma(x_bad, 2, 1) / dgamma(x_bad, 1, 2)
plot(w_bad)
```
weighted_mean(x_bad, w_bad)

## [1] 2.313655

Now our importance sampling estimate is clearly off, and the plot of the importance weights show what is going on: There is one huge weight and couple others which will dominate our mean estimator. We can use weight_plot function to further illustrate the differences between our two approaches. First let’s check the results from our good IS case:

weight_plot(w)
The function `weight_plot` draws four figures. The first one shows 100 largest weights, again illustrating that there are no single draws dominating the sample. On the second figure, the weights are sorted and drawn in increasing order. This figure has mixed uses, sometimes it shows interesting phenomena better than for example a histogram (which often needs some tweaking), whereas in other cases its information value is quite small. Here we see that the weights are distributed quite uniformly. The third figure is often the most important; it shows the variance of the weights computed from successive samples $w_1, \ldots, w_t$, $t = 1, \ldots, n$. If the importance weights have finite variance, this figure should show clear converge towards finite values as is the case here. The final figure shows the effective sample sizes again in a form of running line. The effective sample size (ESS) is defined as

$$ESS_n = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\sum_{i=1}^{n} w_i^2} = \frac{1}{\sum_{i=1}^{n} \bar{w}_i^2},$$

where $\bar{w}_i = w_i / \sum_{i=1}^{n} w_i$. The interpretation of ESS is that our importance sampling corresponds to the case with direct simulation from the target distribution $p(x)$ using ESS samples (i.e. larger the ESS is better). Other measures of efficiency are also available in literature, and they might be added to diagis in future.

Now let’s use `weight_plot` function for our bad IS run:

```r
weight_plot(w_bad)
```
Oops! The figures are pretty self-explanatory, the running statistics look fine at first, but when we take that one huge weight into account, the variance explodes and at the same time ESS drops significantly as that one \((x_i, w_i)\) pair dominates the sums in ESS.

References


Vihola, Matti, Jouni Helske, and Jordan Franks. 2017. “Importance Sampling Type Estimators Based on