Introduction to importance sampling

Matthew Stephens *University of Chicago*

See here for a PDF version of this vignette.

Definition

Suppose we want to compute the expectation of a function $f(X)$ with respect to a distribution with density $p(x)$. So

$$I = \int f(x)p(x)\,dx.$$

That is,

$$I = E[f(X)],$$

where the expectation is taken over a random variable X with density p. We could approximate I by "naive simulation":

$$I \approx \frac{1}{M}\sum_{m=1}^{M} f(X_m),$$

where $X_1, \ldots, X_M \sim p(x)$.

Now let $q(x)$ denote any other density function that is non-zero whenever $p(x)$ is non-zero. (We need this condition to avoid dividing by zero in what follows.) Then we can rewrite this as

$$I = \int f(x)\,[p(x)/q(x)]\,q(x)\,dx.$$

That is,

$$I = E[f(X)p(X)/q(X)]$$

where the expectation is taken over a random variable X with density q. So we could also approximate I by simulation:

$$I \approx \frac{1}{M}\sum_{m=1}^{M} f(X'_m) \times \frac{p(X'_m)}{q(X'_m)},$$

where $X'_1, \ldots, X'_m \sim q(x)$.

This is called "Importance Sampling" (IS), and q is called the "importance sampling function".

If q is well chosen, then the approximation to I will be better than the naive approximation.

Examples

Suppose $X \sim N(0,1)$, and we want to compute $\Pr(X > z)$, for $z = 10$. That is, $f(x) = \mathbb{I}(x > 10)$, and $p(x) = \phi(x)$ is the density of the standard normal distribution.

Let's try naive simulation, and compare it with the "truth", as ascertained by the R function "pnorm".

```
set.seed(100)
x <- rnorm(100000)
mean(as.numeric(x > 10))
# [1] 0
pnorm(10,lower.tail = FALSE)
# [1] 7.62e-24
```

You can see the problem with naive simulation: all the simulations are less than 10 (where $f(x) = 0$), so you don't get any precision.

Now we use IS. Here we code the general case for z, using IS function q to be $N(z, 1)$. Note that because of this choice of q many of the samples are greater than z, where f is non-zero, and we hope to get better precision. Of course, we could solve this problem in much better ways... this is just a toy illustration of IS. [Hint: note that `mean(w*(y > z))` is the same as `mean(y > z) * mean(w[y > z])`.]

```
# Return IS estimate of integral, with IS function N(0,1).
pnorm_IS <- function (z, nsamp = 100000) {
  y <- rnorm(nsamp,z,1)
  w <- exp(dnorm(y,0,1,log = TRUE) - dnorm(y,z,1,log = TRUE))
  return(mean(w*(y > z)))
}
pnorm_IS(10)
# [1] 7.674e-24
pnorm(10,lower.tail = FALSE)
# [1] 7.62e-24
```

Example: computing with means on the log scale

To push this example a bit further, let's illustrate a numerical issue that can arise quite generally (not just in IS).

Let's try the above with $z = 100$.

```
pnorm_IS(100)
# [1] 0
pnorm(100,lower.tail = FALSE)
# [1] 0
```

Hmm... we are having numerical issues.

The trick to solving this is to do things on the log scale. But the IS formula involves averaging, and we have to do the averaging on the raw scale, not the log scale. Here is a function that allows us to do this. Perhaps you can work out what is going on here?

```
# Return the log of the mean of exp(lx).
lmean <- function (lx) {
```

```
  m <- max(lx)
  return(m + log(mean(exp(lx - m))))
}
```

Exploiting this function, we can now do IS for $z = 100$:

```
lpnorm_IS <- function (z,nsamp = 100000) {
  y <- rnorm(nsamp,z,1)
  lw <- dnorm(y,0,1,log = TRUE) - dnorm(y,z,1,log = TRUE)
  return(log(mean(y > z)) + lmean(lw[y > z]))
}
lpnorm_IS(100)
# [1] -5006
pnorm(100,lower.tail = FALSE,log = TRUE)
# [1] -5006
```