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":

$$I \approx (1/M)\sum_m f(X_m)$$

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 0 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:

$$I \approx (1/M)\sum_m f(X'_m)p(X'_m)/q(X'_m)$$

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

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

The idea behind IS is that 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, $f(x) = I(x > 10)$ 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(1*(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 $> z$, where f is non-zero, and we hope to get better precision. Of course, we could do this problem much better ways... this is just a toy illustration of IS.

```
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))
  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 log scale

We just push this example a bit further, to illustrate a numerical issue that can arise quite generally (not just for IS).

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

```
pnorm.IS(100)
```

```
## [1] 0
```

```r
pnorm(100,lower.tail=FALSE)
```

```
## [1] 0
```

Hmmm.. we are having numerical issues.

The trick to solving this is doing things on log scale. But the IS formula involves averaging, and we have to do the averaging on the raw scale, not the log scale. This function allows us to do this. Perhaps you can work out what is going on here? [Hint: note that `mean(w*(y>z)) = mean(y>z) * mean(w[y>z])`.]

```r
#function to find the log of the mean of exp(lx).
lmean=function(lx){
  m = max(lx)
  m + log(mean(exp(lx-m)))
}
```

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

```r
lpnorm.IS= function(z,nsamp=100000){
  y = rnorm(nsamp,z,1)
  lw = dnorm(y,0,1,log=TRUE) - dnorm(y,z,1, log=TRUE) # log-weights
  log(mean(y>z)) + lmean(lw[y>z])
}
lpnorm.IS(100)
```

```
## [1] -5006
```

```r
pnorm(100,lower.tail=FALSE,log=TRUE)
```

```
## [1] -5006
```