统计计算 张楠 2019秋: Variance Reduction: Importance Sampling

(返回 统计计算 张楠 2019秋)

Importance Sampling

在前面内容中,我们将有限区间上的积分视为是此区间上的均匀分布随机变量的某个函数的期望值. 这种方法的缺点是不能直接用于无穷积分的估计, 以及当被积函数在积分区间上不是很均匀的时候 效率很低. 那么很自然地可以考虑均匀分布以外的其他加权函数. 这就导致"重要性抽样方法".

假设随机变量X的密度函数为$f(x)$, 满足在集合$\{x : g(X) > 0\}$上$f(x) > 0$. 则记$Y = g(X)/f(X)$,

$$\int g(x)dx = \int \frac{g(x)}{f(x)} f(x)dx = EY.$$

通过简单的Monte Carlo方法估计EY:

$$\frac{1}{m}\sum_{i=1}^{m} Y_i = \frac{1}{m}\sum_{i=1}^{m} \frac{g(X_i)}{f(X_i)}.$$

此处随机变量$X_1, \cdots, X_m$为从f中抽取的样本. f称为;**重要函数**. 估计的 方差为$Var(Y)/m$, 当Y接近常数时, Y的方差会比较小, 此时密度f应该``靠近" 函数$g(x)$. 当然, 密度f应该选择容易抽样的分布.

> 重要性抽样方法的优点是可以通过选择*重要函数*来降低 Monte Carlo 估计的方差. 假设$f(x)$是支撑为A的密度函数, 若$\phi(x) > 0, \ \forall x \in A$, 则积分
> $\theta = \int_A g(x)f(x)dx,$

可以被写为

$$\theta = \int_A g(x)\frac{f(x)}{\phi(x)}\phi(x)dx.$$

若$\phi(x)$为A上的密度函数, 则θ的一个估计量为

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} g(X_i)\frac{f(X_i)}{\phi(X_i)},$$

其中$X_1, \cdots, X_n$为从$\phi(x)$中产生的随机样本. $\phi(x)$称为*重要性抽样函数*或者 *包络函数*(envelope). ϕ的选择有各种各样的, 典型的选择使得在A上有 $\phi(x) \approx |g(x)|f(x)$, 当然$\phi$有有限的方差. 这是由于

$$Var(Y) = \int_A \frac{g^2(x)}{\phi(x)}dx - \theta^2.$$

由Cauchy-Schwarz不等式, $Var(Y)$的最小值为

$$\left(\int_A |g(x)|dx\right)^2 - \theta^2.$$

可以通过取

$$\phi(x)=\frac{|g(x)|}{\int_A |g(x)|dx}.$$

不过由于要估计$\int_A |g(x)|dx$, 因此分母未知. 尽管这样能达到最小方差的$\phi(x)$很难取到, 但是如果选择和$|g(x)|$形状接近的$\phi(x)$, 则方差可以接近最优方差.

例10: 重要性函数的选择 使用不同的重要性函数来估计积分 $\int_0^1 \frac{e^{-x}}{1+x^2}dx.$

比如, 可以考虑如下重要性函数:
$$\begin{aligned} f_0(x) &= 1,\ 0<x<1, \\ f_1(x) &= e^{-x},\ 0<x<\infty, \\ f_2(x) &= \frac{1}{\pi(1+x^2)},\ -\infty<x<\infty, \\ f_3(x) &= e^{-x}(1-e^{-1})^{-1},\ 0<x<1, \\ f_4(x) &= \frac{4}{\pi(1+x^2)},\ 0<x<1 \end{aligned}$$
这5种函数和g之间的关系:

文件:Stat comp lec5 pic1.pdf

文件:Stat comp lec5 pic2.pdf

在这5种重要性函数中, f_1, f_2的支撑比$(0,1)$大, 因此很多样本都对估计积分无用, 因此效率低下. f_3最接近g. 比较这5种重要性函数下的积分估计值和方差估计值:

```
m <- 10000
theta.hat <- se <- numeric(5)
g <- function(x) {
    exp(-x - log(1+x^2)) * (x > 0) * (x < 1)
    }

x <- runif(m)      #using f0
fg <- g(x)
theta.hat[1] <- mean(fg)
se[1] <- sd(fg)

x <- rexp(m, 1)    #using f1
fg <- g(x) / exp(-x)
theta.hat[2] <- mean(fg)
se[2] <- sd(fg)

x <- rcauchy(m)    #using f2
i <- c(which(x > 1), which(x < 0))
x[i] <- 2  #to catch overflow errors in g(x)
fg <- g(x) / dcauchy(x)
theta.hat[3] <- mean(fg)
se[3] <- sd(fg)

u <- runif(m)      #f3, inverse transform method
x <- - log(1 - u * (1 - exp(-1)))
fg <- g(x) / (exp(-x) / (1 - exp(-1)))
theta.hat[4] <- mean(fg)
se[4] <- sd(fg)

u <- runif(m)     #f4, inverse transform method
x <- tan(pi * u / 4)
fg <- g(x) / (4 / ((1 + x^2) * pi))
theta.hat[5] <- mean(fg)
se[5] <- sd(fg)

rbind(theta.hat, se)
```