Marginal likelihood computation comparison

Naveed Merchant

October 12, 2018

We first generate data and check what our marginal likelihood's should be by the function given to us:

```
set.seed(500)
dataset<- rnorm(500)
X1 <- dataset[1:(length(dataset)*.3)]
X2 <- dataset[-(1:(length(dataset)*.3))]
B <- unname((quantile(X2,probs = .75) - quantile(X2,probs = .25))/1.35)
k <- length(X1)
n <- length(X2)
loglike.KGauss=function(h,y,x){
  n=length(x)

  m=length(y)

  nh=length(h)

  M=t(matrix(y,m,n))

  llike=1:nh

  for(j in 1:nh){

    M1=(x-M)/h[j]

    M1=dnorm(M1)/h[j]

    fhat=as.vector(M1 %*% matrix(1,m,1))/m

    fhat[fhat<10^(-320)]=10^(-320)

    llike[j]=sum(log(fhat))

  }

  -llike

}



integrand.Gauss=function(h,y,x,cons,prior){

  n=length(x)

  R=quantile(y,probs=c(.25,.75))

  R=R[2]-R[1]
```

```r
  beta=R/1.35

  beta1=beta*log(2)/sqrt(qgamma(.5,.5,1))

  Prior=beta1*exp(-beta1/h)/h^2

  if(prior==1) Prior=(2*beta/sqrt(pi))*h^(-2)*exp(-beta^2/h^2)

  arg=-loglike.KGauss(h,y,x)-cons

  arg[arg>700]=700

  f=exp(arg)*Prior

  f

}
logmarg.kern=function(y,x,prior=1){

  out=optim(.4,loglike.KGauss,method="L-BFGS-B",lower=.0001,upper=5,y=y,x=x)

  h=out$par

  cons=-out$val

  stat=integrate(integrand.Gauss,lower=0.0001,upper=5,y=y,x=x,cons=cons,prior=prior)$val

  list(h,cons+log(stat))

}
#Code supplied by Dr. Hart
logmarg.kern(X1,X2,prior=1)
```

```
## [[1]]
## [1] 0.3626156
##
## [[2]]
## [1] -512.5608
```

Log of the integral is roughly -512.

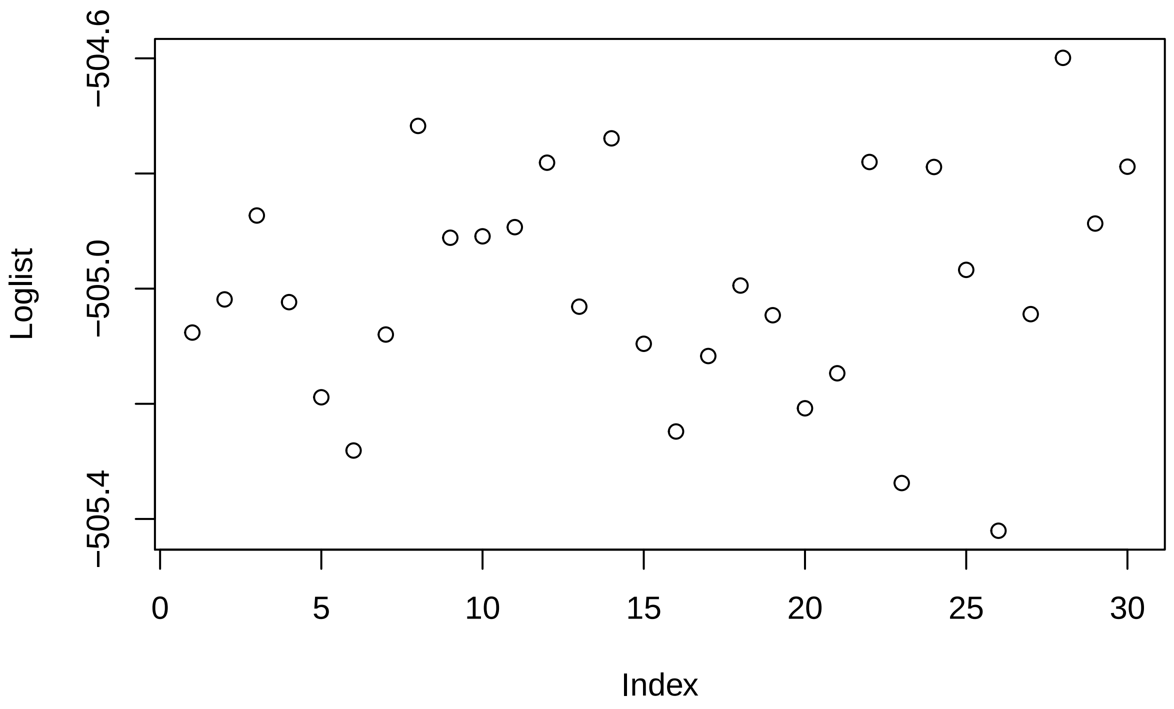
We compare this to the method we've normally been doing (importance sampling). Sample positive truncated cauchy. Then compute likelihood, using samples from positive truncated cauchy as our guesses for h. As this is from a cauchy distribution and not the prior, we multiply by a small term to ensure that our result isn't biased. This gives us a guess for the integral. We repeat this procedure many times and average the results. The expected value of the average should be equal to the integral.

As this is a random method, we repeat this method 50 times to examine the variation between guesses of the integral

```r
Loglist <- c()
for(G in 1:30)
{

  cauchsamp<- rcauchy(500)
  poscauchsamp <- cauchsamp[cauchsamp > 0]

  importancepart <- ((2*B*(1/poscauchsamp^2)*(exp(-(B^2)/(poscauchsamp)^2)) / sqrt(pi))) / (2*dcauchy(po

  prodlist1<-c()

  for(z in 1:length(poscauchsamp))
  {
    prod <- 0
    for(j in 1:n)
    {
      sum <- 0
      for(i in 1:k)
      {
        sum <- sum + exp(-.5*((X2[j]-X1[i])/poscauchsamp[z])^2)
      }
      prod <- prod + log(sum) + log((1/sqrt(2*pi))) - log((k*poscauchsamp[z]))
    }
    prodlist1[z] <- prod + log(importancepart[z])
  }
  Loglist[G] <- logSumExp(prodlist1)
}
summary(Loglist)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -505.4  -505.1  -505.0  -505.0  -504.9  -504.6
```

```
plot(Loglist)
```



This isn't quite the same, not only is the median / mean of the importance samples not the same as the integral, the variance isn't large, and our guesses of the marginal likelihood are centering far away from where the integral should be...

I don't think I'm doing importance sampling incorrectly either...

I noticed an error in the PDF I sent you, the expression of the integral inside the sum is actually: $IG(\frac{n-1}{2}, .5(2B^2 + \sum_{L=1}^{n}(X_{2,L} - X_{1,i_L})^2))$

Not

$IG(\frac{n+3}{2}, .5(2B^2 + \sum_{L=1}^{n}(X_{2,L} - X_{1,i_L})^2))$

This ends up not changing the integral by too much, but it's still worth a note.

We examine the third way we discussed. We know the integral appears like:

$(\frac{1}{k\sqrt{2\pi}})^n \frac{B}{\sqrt{\pi}} \sum_{i_n=1}^{k} \sum_{i_{n-1}=1}^{k} \cdots \sum_{i_2=1}^{k} \sum_{i_1=1}^{k} \frac{\Gamma(\frac{n-1}{2})}{(.5(2B^2+\sum_{L=1}^{n}(X_{2,L}-X_{1,i_L})^2))^{\frac{n-1}{2}}}$

So we compute $\frac{\Gamma(\frac{n-1}{2})}{(.5(2B^2+\sum_{L=1}^{n}(X_{2,L}-X_{1,i_L})^2))^{\frac{n-1}{2}}}$, many times and average over the number of times we computed it.

We don't know i_l but we do know for it is an integer between 1 to k, and i_l is equally often each integer. So we randomly uniformally sample an integer from 1 to k, pretend that to be i_l, and compute the average of those guesses. We expect this to be close to

$(\frac{1}{k})^n \sum_{i_n=1}^{k} \sum_{i_{n-1}=1}^{k} \cdots \sum_{i_2=1}^{k} \sum_{i_1=1}^{k} \frac{\Gamma(\frac{n-1}{2})}{(.5(2B^2+\sum_{L=1}^{n}(X_{2,L}-X_{1,i_L})^2))^{\frac{n-1}{2}}}$

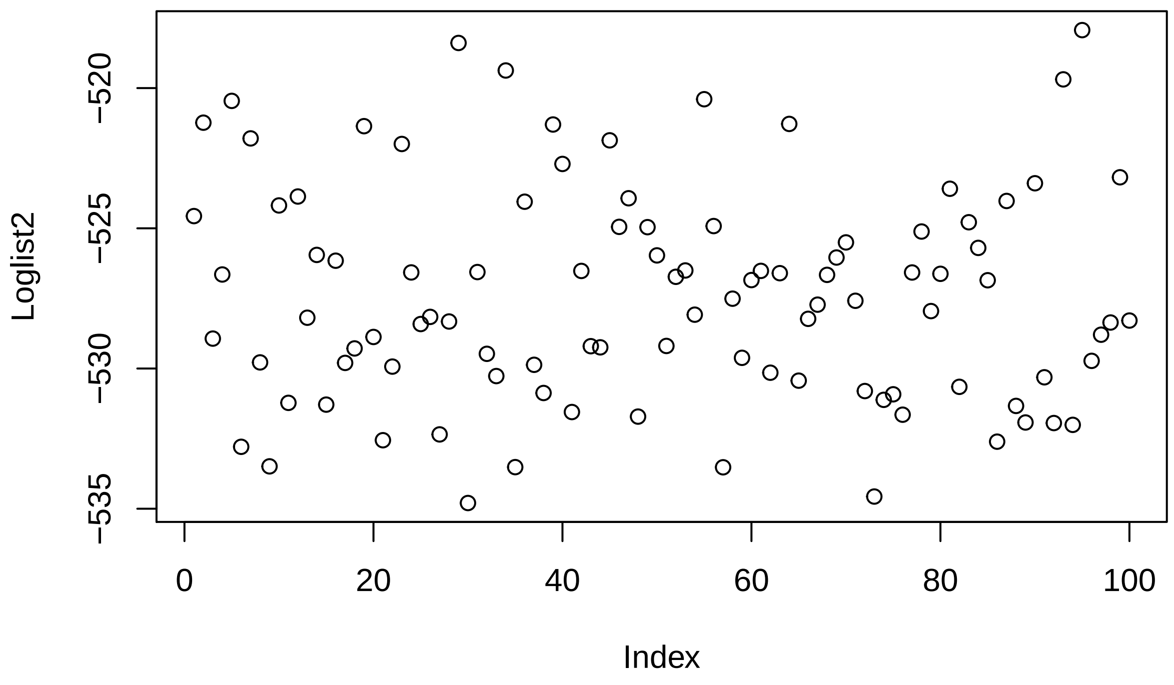
So all we have to do is multiply this result by $(\frac{1}{\sqrt{2\pi}})^n \frac{B}{\sqrt{\pi}}$ and we should have a decent idea of what the integral should be.

This is supposed to be significantly faster than importance sampling.

```
Loglist2 <- c()
for(K in 1:100)
{
  sum1list <- c()
  iter <- 10000
  for(i in 1:iter)
  {
    l <- sample(1:k,n, replace = TRUE)
    sum1 <- sum((X2[j] - X1[l])^2)
    sum1 <- lgamma((n-1)/2) - ((n-1)/2)*log(.5*(2*B^2 + sum1))
    sum1list[i] <- sum1
  }
  logMCinteg <-log(B / sqrt(pi)) - (n/2)*log(2*pi) + logSumExp(sum1list) -log(iter)
  Loglist2[K] <-  logMCinteg
}
summary(Loglist2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -534.8  -530.3  -527.8  -527.4  -524.9  -517.9
```

```
plot(Loglist2)
```



The guess for this integral is much further away then what the previous results were and quite variable as well.

I think I may have made more mistakes in derivation, as that could imply why the results of the MC are so far off…

Or something else is wrong.

In hindsight doing importance sampling is a silly idea for a 1-dimensional integral too. I think sampling was a good idea in very high dimensional problems because the size of the sample determines the variance and not the size of the dimension.

Should I just rely on your function for computing the marginal likelihood?

“‘