Log of the Marginal likelihood when Permuted Examination

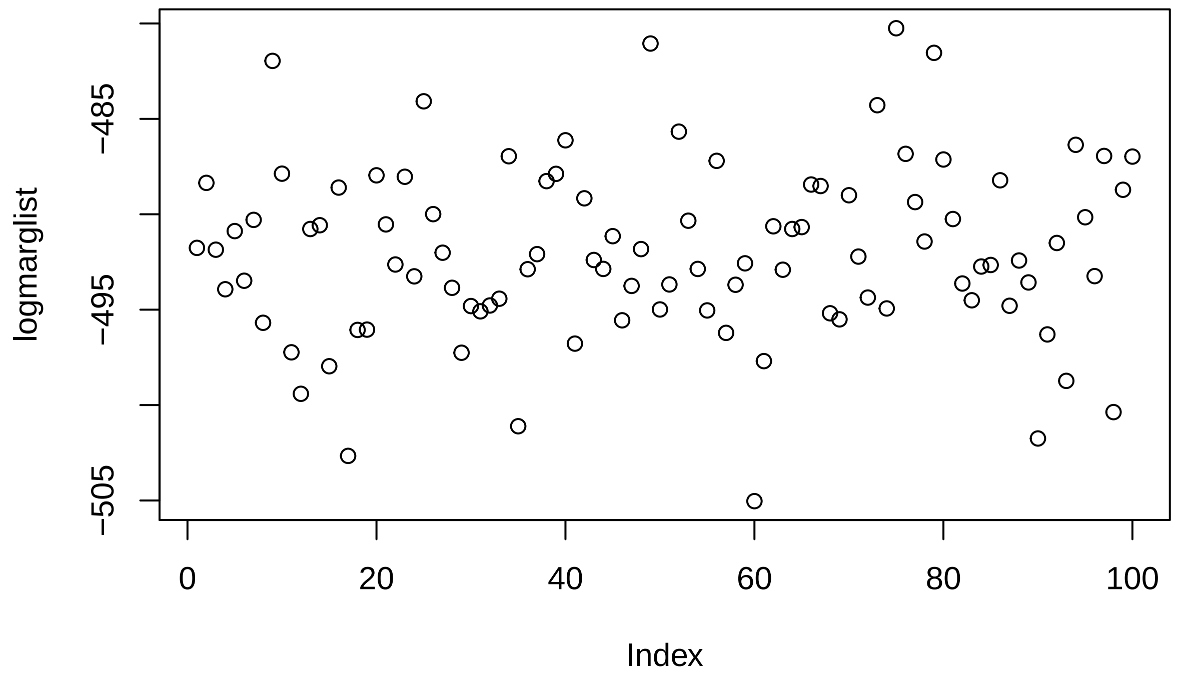
Naveed Merchant

October 15, 2018

We draw data, permute it, grab a portion of the samples, and examine what the marginal likelihood looks under it.

To get the marginal likelihood, we rely on the function we've been given

```
dataset<- rnorm(500, mean = 10, sd = 1)
source("MarginalLikIntfunctions.R")
iter <- 100
logmarglist <- c()
for(i in 1:iter)
{
  dataset2 <- sample(dataset)
  X1 <- dataset2[1:(length(dataset)*.3)]
  X2 <- dataset2[-(1:(length(dataset)*.3))]
  logmarglist[i] <- logmarg.kern(X1,X2,prior = 1)[[2]]
}
plot(logmarglist)
```



```r
summary(logmarglist)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -505.0  -494.8  -492.3  -492.0  -488.7  -480.3
```

This takes a fair bit longer than you'd expect.

The function takes a reasonably long amount of time to run I suppose.

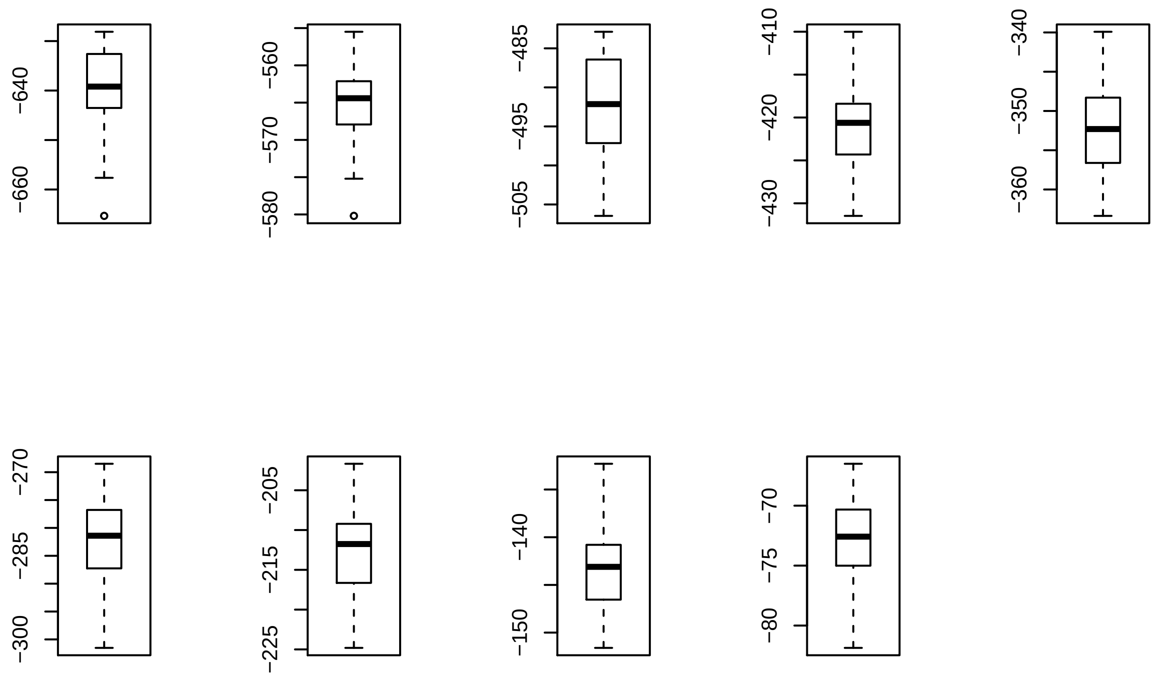
Regardless it seems that there's quite a bit of variability in the different types of samples that can be drawn.

It is suggested that this sort of idea is harshly dependent on the porportion of samples being used for training. We vary the proportion from .1 - > .9

```r
p <- seq(from = .1, to = .9, by = .1)
iter <- 50
logmarglist2 <- list()
for(j in 1:length(p))
{
  logmarglist2[[j]] <- vector(mode  = "double", length = iter)
  for(i in 1:iter)
  {
    dataset2 <- sample(dataset)
    X1 <- dataset2[1:(length(dataset)*p[j])]
    X2 <- dataset2[-(1:(length(dataset)*p[j]))]
    logmarglist2[[j]][i] <- logmarg.kern(X1,X2,prior = 1)[[2]]
  }
  print(j)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
```

```r
par(mfrow = c(2,5))
for(i in 1:length(logmarglist2))
{
  boxplot(logmarglist2[[i]])
}
```



This is alarming!

We have to be very careful, the marginal likelihood harshly depends on the choice of p we use (in hindsight this is not actually suprising. . .).

Basically if we make a test we either have to take into consideration that if we are training we are

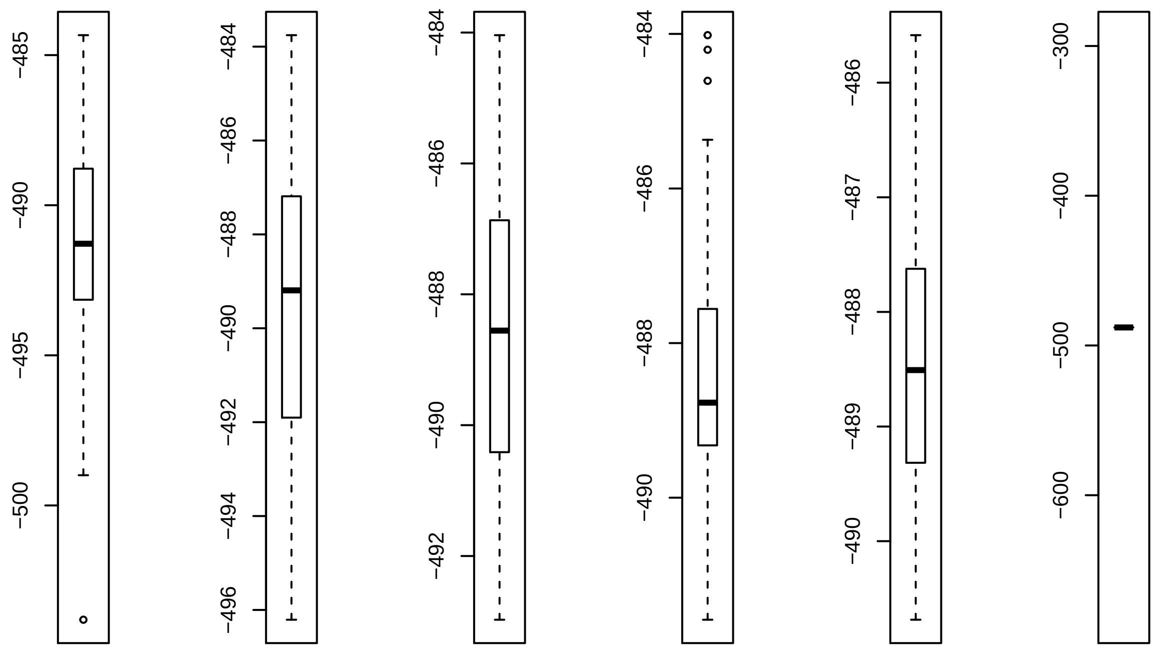
```
datasetT <- dataset[1:(length(dataset)*.3)]
datasetV <- dataset[(length(dataset)*.3+1):length(dataset)]

p <- seq(from = .5, to = 1, by = .1)
iter <- 50
logmarglist3 <- list()

for(j in 1:length(p))
{
  logmarglist3[[j]] <- vector(mode  = "double", length = iter)
  for(i in 1:iter)
  {
    dataset2 <- sample(datasetT)
    X1 <- dataset2[1:(length(datasetT)*p[j])]
    logmarglist3[[j]][i] <- logmarg.kern(X1,datasetV,prior = 1)[[2]]
  }
  print(j)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

```r
par(mfrow = c(1,6))
for(i in 1:length(logmarglist3))
{
  boxplot(logmarglist3[[i]])
}
```



So this is a good sign.

It appears that fixing the size of our validation set has prevented the log marginal likelihood from changing harshly in magnitude. So we can use this. We fix our validation set, and use different sized training sets. It appears as if their marginal likelihoods would be similar which is good.

So my thoughts are the following. Suppose we have some data from 2 distributions that are drawn respectively from function f and g. We want to check if f = g. Split f and g into testing and validation data sets. We then do the following. Take testing dataset for f and compute ML for it with f's validation. Do the same but use f and g's testing datasets to make a big testing dataset, use f's validation set as the validation set and see how it does.

We repeat the same idea but with g's validation set being used instead.

Will run simulations on this idea later.

“‘