STA 138 Discussion 5 – solutions

Fall 2020

Maximum likelihood estimation by numerical methods

For our discussion this week, we will further explore maximum likelihood estimation with numerical methods.

MLE under the multinomial model

For our multinomial model with c categories, remember that the parameters $\pi_1, \ldots, \pi_c$ have the constraint

$$\sum_{j=1}^{c} \pi_j = 1 \ .$$

Thus, we write any one of these as a function of the others, for example

$$\pi_1 = 1 - \sum_{j=2}^{c} \pi_j$$

so that the likelihood can be considered to be a function of $c-1$ freely varying parameters:

$$L(\pi_1, \ldots, \pi_c, y_1, \ldots, y_c) = \frac{n!}{\prod_{j=1}^{c} y_j!} \left(1 - \sum_{j=2}^{c} \pi_j\right)^{y_1} \prod_{j=2}^{c} \pi_j^{y_j}$$

It is possible to maximize this function with respect to $\pi_2, \ldots, \pi_c$ analytically, but as we'll see, other likelihood functions can get somewhat complicated. Therefore, it's useful to see how we can do this numerically.

First, let's save our counts:

```
counts <- c(14,22,25,15)
```

Let's then represent the likelihood function itself:

```
multinomLik <- function(pi, y=counts){
    dmultinom(y, prob = pi)
}
```

It will be important that we keep track of which parameters vary freely. To do this, let's write a function that maps some freely varying values to a full vector of c parameters (let's write π_1 as a function of the freely varying values $\pi_j, j = 2, \ldots, c$):

```
allPi <- function(freePi){
   c(1-sum(freePi), freePi)
   }
```

This function takes vector $(\pi_2, \ldots, \pi_c)$ as an argument and returns the full vector $(\pi_1, \ldots, \pi_c)$.

It only remains for us to maximize this function over possible freely varying values π_j. One way to do this is with `optim` which, by default, uses the Nelder-Mead algorithm to iteratively approximate the minimim of a function. Other methods are possible, but this is convenient here because, for example, it does not require an explicit representation of the gradient of our likelihood function. Like most iterative methods, this does require an initial value; we will use the uniform choice $\pi_2 = 1/4, \pi_3 = 1/4, \pi_4 = 1/4$ for this.

```
init <- rep(0.25,3)
optOutput <- optim(init,
       fn = function(freePi){ -multinomLik( allPi( freePi ) ) }
       )
```

we need to inspect the results in order to make sure that this has performed as required:

```
optOutput$convergence
```

```
[1] 0
```

This suggests (we can check the `optim` help file) that the algorithm has converged. The MLE obtained here is given by:

```
allPi(optOutput$par)
```

```
[1] 0.1842099 0.2894753 0.3289473 0.1973674
```

It's instructive to compare these to the sample proportions y_j/n :

```
counts/sum(counts)
```

```
[1] 0.1842105 0.2894737 0.3289474 0.1973684
```

so the MLE is almost identical to the sample proportions (the small difference is actually just due to numerical error).

Let's note in passing that we can write the maximum value of the likelihood itself as well:

```
multinomLik(allPi(optOutput$par))
```

```
[1] 0.001600415
```

This number is difficult to interpret in itself, but we'll see that it comes in handy down the road, in things like computing likelihood ratio test statistics.

Constrained estimation

One of the nice features of maximum likelihood estimation is that it adapts in a simple way to additional constraints. This is sometimes useful to estimation in special cases, but more importantly (as we'll see moving forward), it comes in handy for evaluating claims about values of subsets of the possible parameters.

Suppose that we want to evaluate the claim that $\pi_1 = 0.2$ and that $\pi_2 = 0.3$. Let's assume that this is true. What should be the MLE for $\pi = (\pi_1, \ldots, \pi_4)$ be with this additional constraint?

We have now the same likelihood function as before. However, now there are now *three* constraints on π:

$$\text{(i)} \ \sum_{j=1}^{c} \pi_j = 1$$
$$\text{(ii)} \ \pi_1 = 0.2$$
$$\text{(iii)} \ \pi_2 = 0.3$$

and so there is only one freely varying parameter. Together, these impose on us that

$$\pi_3 + \pi_4 = 0.5 \ ,$$

and hence

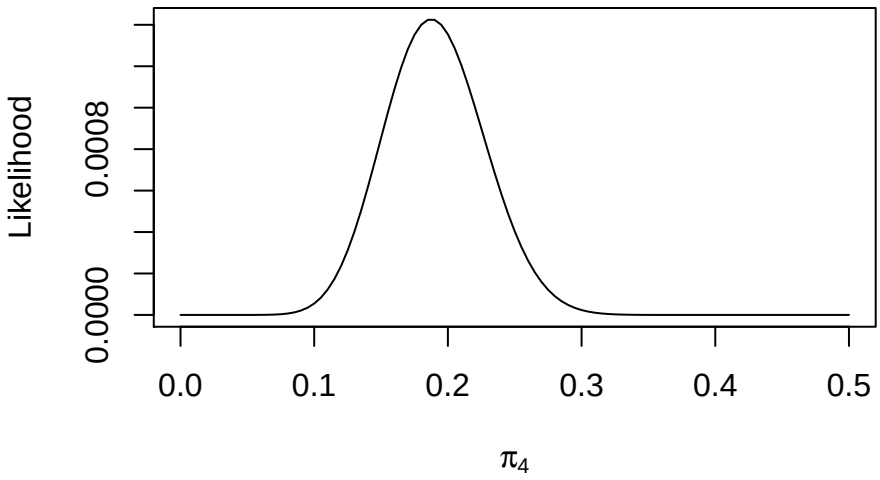
$$\pi_3 = 0.5 - \pi_4 \ .$$

Thus:

```
allPi2 <- function(freePi){
   c( 0.2, 0.3, 0.5-freePi, freePi)
   }
```

In this case, with only one freely varying parameter, it is possible to plot the graph of the likelihood function easily:

```
plot(function(x){ sapply(x,function(z){multinomLik( allPi2(z) )}) },
     from=0,
     to=0.5,
     xlab=bquote(pi[4]),
     ylab="Likelihood")
```



For one dimentional optimization, optmimize is recommended over optim:

```
optOutput2 <- optimize( f = function(freePi){ -multinomLik( allPi2( freePi ) ) },
                        interval = c(0,0.5)
)
```

With the additional contraints, we should expect that the maximal likelihood is smaller:

```
multinomLik(allPi2(optOutput2$minimum))
```

```
[1] 0.001427126
```

as indeed it is.

Furthermore, we can compare the MLE itself under these constraints to our earlier one:

π_1	π_2	π_3	π_4
0.1842099	0.2894753	0.3289473	0.1973674
0.2000000	0.3000000	0.3124894	0.1875106

The constrained MLE, given in the bottom row in the table above, admits a natural interpretation. The constraints $\pi_1 = 0.2$ and $\pi_2 = 0.3$ are slightly larger than the respective sample proportions. Therefore, the MLE under these constraints estimates π_3 and π_4 to be slightly smaller than it would have otherwise to compensate.

```
counts <- c(14,22,25,15)
multinomLik <- function(pi, y=counts){
    dmultinom(y, prob = pi)
}
allPi <- function(freePi){
   c(1-sum(freePi), freePi)
   }
init <- rep(0.25,3)
optOutput <- optim(init,
       fn = function(freePi){ -multinomLik( allPi( freePi ) ) }
       )
optOutput$convergence
allPi(optOutput$par)
counts/sum(counts)
multinomLik(allPi(optOutput$par))
allPi2 <- function(freePi){
   c( 0.2, 0.3, 0.5-freePi, freePi)
   }
plot(function(x){ sapply(x,function(z){multinomLik( allPi2(z) )}) },
     from=0,
     to=0.5,
     xlab=bquote(pi[4]),
     ylab="Likelihood")
optOutput2 <- optimize( f = function(freePi){ -multinomLik( allPi2( freePi ) ) },
                        interval = c(0,0.5)
)
multinomLik(allPi2(optOutput2$minimum))
library(knitr)
M <- rbind(allPi(optOutput$par),allPi2(optOutput2$minimum))
colnames(M)<- paste("$\\pi_",1:4,"$",sep="")
kable(M)
```