A weighted self-concordant optimization for empirical likelihood

Art B. Owen

February 2017

Abstract

The empirical likelihood code in `scel.R` has been modified to take counts. That way if the value $z_i \in \mathbb{R}^d$ appears c_i times we don't need to put in c_i copies of z_i.

Function redefinition

The new function definition is

```
function(  z,   # matrix with one data vector per row, a column vector is ok when d=1
          ct,   # vector of counts
          mu,   # hypothesized mean, default (0 ... 0) in R^d
         lam,   # starting lambda, default (0 ... 0)
         eps,   # lower cutoff for -log( ), default 1/nrow(z)
           M,   # upper cutoff for -log( ), default Inf
thresh=1e-30,   # convergence threshold for log likelihood (default is aggressive)
 itermax=100,   # upper bound on number of Newton steps (seems ample)
 verbose=FALSE) # controls printed output
```

and the function has been renamed from `emplik` to `cemplik`. At this point, the old function was copy-pasted and edited to handle weights. That brings some technical debt. If the new version holds up to testing, I might replace the old one by the new one with default weights all equal to 1.

The counts do not have to be natural numbers. As such they are treated as weights. Non-integer values may be useful but at this point no statistical interpretation. If one wants to implement Horvitz-Thompson weighting, for importance sampling or for propensity scoring, then that weighting can be built in to the estimating equations.

Weights $c_i = 0$ are convenient for temporarily removing some data points from the sample.

The way the likelihood is optimized (Newton's method using a least squares formulation) makes weights c_i near to zero inconvenient. The code does not allow them. The weights must satisfy $c_i \in \{0\} \cup [0.001, \infty)$ and $\sum_i c_i > 0$.

One approach to escaping the convex hull is to put in an unobserved data point that makes the hypothesized mean belong to the convex hull of the expanded data set. Such a point could possibly be given weight $c_{n+1} = 0.001$. Many authors actually give such points equal weight with the real ones. If one gives it zero weight a different algorithm is needed from the one described below.

Likelihood

The data are $z_i \in \mathbb{R}^d$ which appears c_i times for $i = 1, \ldots, n$. The total sample size is $N = \sum_i c_i$. The empirical distribution function is $\hat{F} = (1/N)\sum_{i=1}^n c_i \delta_{z_i}$. We consider instead the distribution F which places probability w_i on z_i where $w_i \geq 0$ and $\sum_{i=1}^n w_i = 1$.

The likelihood is then $L(F) = \prod_i w_i^{c_i}$ and the likelihood ratio is $R(F) = \prod_i (N w_i/c_i)^{c_i}$. Then $-2\log(R(F)) = -2\sum_i c_i \log(N w_i/c_i)$. To maximize empirical likelihood subject to the constraint $\sum_i w_i z_i = \mu$ we use the Lagrangian

$$G = \sum_i c_i \log(N w_i) - N\lambda^{\mathsf{T}} \sum_i w_i (z_i - \mu) + \delta\left(\sum_i w_i - 1\right).$$

The term $\sum_i c_i \log(c_i)$ does not affect the optimization over w_i and so we omit it. In the limit $c_i \to 0$, such a term approaches zero. Now

$$\frac{\partial G}{\partial w_i} = \frac{c_i}{w_i} - N\lambda^{\mathsf{T}}(z_i - \mu) + \delta.$$

Multiplying by w_i and summing over i we get $\delta = -N$. Then

$$w_i = \frac{c_i}{N}\frac{1}{1 + \lambda^{\mathsf{T}}(z_i - \mu)}.$$

These are the usual empirical likelihood weights multiplied by c_i. Next we form the dual likelihood

$$L(\lambda) = \prod_{i=1}^n \Big(\frac{w_i}{c_i/N}\Big)^{c_i} = \prod_{i=1}^n \big(1 + \lambda^{\mathsf{T}}(z_i - \mu)\big)^{c_i},$$

and the dual log likelihood

$$\ell(\lambda) = \sum_{i=1}^n c_i \log(1 + \lambda^{\mathsf{T}}(z_i - \mu)).$$

From this, we see that to modify the empirical likelihood algorithm we have to multiply the contribution to ℓ from observation i by c_i and we also make similar linear adjustments to the first and second derivatives of ℓ.

Code changes

Most of the code is made up of internal functions, especially `mllog` which computes $-\log_\star$ and one or two of its derivatives. The function $\log_\star$ equals log inside the interval $[\epsilon, M]$ and uses fourth order Taylor approximations to log outside of that interval. The modification to the code is then to scale the columns of `mllog` by `ct`.

Simply scaling the columns by `ct` brings a problem when some $c_i = 0$. The Newton step in empirical likelihood is computed by a least squares computation. For $c_i = 0$ we get `NaN` values when we encode the Newton step via least squares. It is very valuable to retain the least squares formulation because it can be done via SVD which has better numerical conditioning than one gets by solving the normal equations.

The solution in the revised algorithm is to eliminate the $c_i = 0$ terms from $\ell(\lambda)$ above and then use least squares. The code does not allow weights below a tolerance (e.g., 0.001) so that there will be a clear distinction between points that are included versus excluded. That tolerance is implemented by the variable `wttol` which is *not* a calling line argument. If anybody wants to tweak it, they will have to do some testing.

The weights used to be computed via $(1/n)/(1 + \lambda^{\mathsf{T}}(z_i - \mu))$. Now they are $(c_i/\sum_j c_j)/(1 + \lambda^{\mathsf{T}}(z_i - \mu))$.

Tests

The revised code was tested against the prevous empirical likelihood code. The first test simply take a data set x of 25 observations in 4 dimensions and computes EL via old code and new code with all $c_i = 1$. It gets the same λ, and the same weights.

The second test takes those 25 observations and repeats the first 10 of them and then the first 5 for a total of 40 observations. It then computes the EL with those 40 observations and recomputes it again on 25 observations with 5 counts of 2, 5 counts of 1 and the remaining counts of 1. The same log empirical likelihood arises and the same Lagrange multipliers too (up to numerical precision).

The third test takes 50 observations from the unit square to test whether the first 49 of them have mean $(1/2, 1/2)$. It does this via EL on those first 49 observations and then again by counted EL with a zero count for the 50'th observation. Since the test point $(1/2, 1/2)$ is inside the convex hull of the first 49 observations that 50th point should get weight 0.

Test output

```
Makes some tests and assigns output to the calling environment.

Example 1, all counts = 1, output saved to elx1 and celx1.
Range of weight differences:
[1] 0 0
Difference in lagrange multipliers:
     [,1]
[1,]    0
[2,]    0
[3,]    0
[4,]    0

Example 2, weighted data, output saved to elx2 and celx2.
Logelr unweighted: -2.939131
Logelr weighted:   -2.939131
Difference in lagrange multipliers:
              [,1]
[1,]  0.000000e+00
[2,]  0.000000e+00
[3,]  2.081668e-16
[4,] -2.775558e-17

Example 3, 50 points with last getting weight 0, versus just the first 49 points.
Output saved to elx3 and celx3.
Logelr usual: -1.897753
Logelr zero wt: -1.897753

Range of weight differences: 0 0
```

Raw R code for testing

```
testit = function(seed=20170219){
# Test cemplik using emplik

set.seed(seed)

cat("\nMakes some tests and assigns output to the calling environment.\n")

# Test with all counts = 1
x = matrix(rnorm(100),ncol=4)
n = nrow(x)

elx1 = emplik(x)
celx1 = cemplik(x,ct=rep(1,n))
cat("\nExample 1, all counts = 1, output saved to elx1 and celx1.\n")
cat("Range of weight differences:\n")
print(range(elx1$wts-celx1$wts))
cat("Difference in lagrange multipliers:\n")
print(celx1$lam-elx1$lam)

# Tests with some counts unequal to 1

xx = rbind(x,x[1:10,],x[1:5,])
ct = rep(1,n)
ct[1:10] = ct[1:10]+1
ct[1:5] = ct[1:5]+1

elx2 <<- emplik(xx)
celx2 <<- cemplik(x,ct)
cat("\nExample 2, weighted data, output saved to elx2 and celx2.\n")
cat("Logelr unweighted: ")
cat(elx2$logelr)
cat("\n")
cat("Logelr weighted:   ")
cat(celx2$logelr)
cat("\n")
cat("Difference in lagrange multipliers:\n")
print(celx2$lam-elx2$lam)

# Test with a zero weight.

u = matrix(runif(100),ncol=2)
elx3 <<- emplik(u[1:49,],mu=c(1/2,1/2))
celx3 <<- cemplik(u,ct=c(rep(1,49),0),mu=c(1/2,1/2))
cat("\nExample 3, 50 points with last getting weight 0, versus just the first 49 points.\n")
```

```
cat("Output saved to elx3 and celx3.\n")
cat("Logelr usual: ");cat(elx3$logelr)
cat("\nLogelr zero wt: ");cat(celx3$logelr)
cat("\n")
cat("\nRange of weight differences: ");cat( range(celx3$wts[-50]-elx3$wts))
cat("\n")
}
```