DOBAD Package: EM Algorithm on a Partially Observed Linear Birth-Death Process

Charles Doss

June 2010

In this Sweave vignette, we will do estimation and confidence intervals for the rates in the restricted-immigration BDI model.

Part I

Estimating Rates for Linear Birth-Death Process via the EM Algorithm

We are demonstrating the use of the `DOBAD` package's capability to do Maximum-Likelihood estimation of the rate parameters for a linear Birth-Death-Immigration (BDI) chain, given partial observations, via the Expectation-Maximization (EM) algorithm. Call the chain $\{X(t)\}_{t\geq 0}$, and its birth rate λ and its death rate μ. We fix $\beta \in \mathbb{R}$ and constrain ν, the immigration rate, to be $\nu = \beta\lambda$. We call this model the restricted-immigration model. We will denote $\theta = (\lambda, \mu)$. The observed data is the value of the process at a finite number of discrete time points. That is, for some fixed times $0 = t_0, t_1, \ldots, t_n$, we see the state of the process, $X(t_i)$. Thus the data, D, is 2 parts: a vector of the times t_i, $i = 0, \ldots, n$ and a vector of states at each of those times, s_i, for $i = 0, \ldots, n$ (where $X(t_i) = s_i$. In order to use the EM algorithm, we need to be able to calculate $E(N_T^+|X_0 = a, X_T = b)$, $E(N_T^-|X_0 = a, X_T = b)$, and $E(R_T|X_0 = a, X_T = b)$, where N_T^+ is the number of jumps up in the time interval $[0, T]$, N_T^- is the number of jumps down in the time interval $[0, T]$, and R_T is the total holding time in the interval $[0, T]$ (i.e. $R_T = \sum_{i=0}^{\infty} i d_T(i)$ where $d_T(i)$ is the time spent in state i in the interval $[0, T]$). We do this via the generating functions.

```
> library(DOBAD)
```

We will set up the true parameters and a true chain, and then "observe" it partially, and see how the EM does on that data. First, we set up the true parameters.

```
> ## set.seed(1155)
> ## initstate=4;
> ## T=8;
> ##nn <-  10; ## numobservs
> ## L <- .5
> ## mu <- .6
> ## beta.immig <- 1.2;
> ## dr <- 0.000001; #Need |dr| < |L-mu| always o/w get sqrt(negative). (numerical differ'n)
> ## n.fft <- 1024;
> ## trueParams <- c(L,mu);
> ## names(trueParams) <- c("lambda", "mu")
>
> set.seed(1155)
> initstate=4;
> T=4;
> nn <-  10; ## numobservs
> L <- .55
> mu <- .6
> beta.immig <- 1.2;
> dr <- 0.000001; #Need |dr| < |L-mu| always o/w get sqrt(negative). (numerical differ'n)
> n.fft <- 1024;
> trueParams <- c(L,mu);
> names(trueParams) <- c("lambda", "mu")
>
```

Now we get the "truth" and then observe the "data" as well as calculate some information about both.

```
> ##Get the "data"
> dat <- birth.death.simulant(t=T, lambda=L, m=mu, nu=L*beta.immig, X0=initstate);
> fullSummary <- BDsummaryStats(dat);
```

```
> fullSummary

   Nplus   Nminus Holdtime
 9.00000  8.00000 11.39358

> names(fullSummary) <- c("Nplus", "Nminus", "Holdtime");
> MLEs.FullyObserved <- M.step.SC( EMsuffStats=fullSummary, T=T, beta.immig= beta.immig);
> #MLEs
> ###MLE.FullyObserved are NOT the MLE for the EM, but hopefully close as delta-of-observati
> partialData <- getPartialData( sort(runif(nn,0,T)), dat);
> observedSummary <- BDsummaryStats.PO(partialData); observedSummary;

   Nplus   Nminus Holdtime
5.000000 2.000000 9.738446
```

`observedSummary` is some measure of the information we're missing. The MLE under partial observations aspires to be as close to the MLE if the full data were observed, ie `MLEs.FullyObserved`. Now we run the actual EM algorithm.

The variable `initParamMat` gets good initial values to start with. We begin the EM with those values here; however, we only run two iterations and then we cheat and restart the EM very close to the optimal values (which we have computed ahead of time). The point is that for the confidence intervals to be accurate, or even to compute at all, the estimates must be reasonable, but we also want this vignette to finish relatively quickly. (If the estimates are not close to the MLE, when we try to compute the confidence interval we can try to take a square root of a negative.)

You may (and should!) modify the number of iterations to see the EM actually at work.

```
> ##########RUN EM
>
> iters <- 1;
> tol <- .001;
> #### You should uncomment the following 2 commands and run -- commented for speed, for now
```

```
> ## initParamMat <- getInitParams(numInitParams=1, summary.PO=observedSummary,
> ##                                      T=T, beta.immig=beta.immig,
> ##                                      diffScale=100*dr);
>
> ## EMtime <- system.time(estimators.hist <-
> ##                       EM.BD.SC(initParamMat=initParamMat, M=iters, beta.immig=beta.immig
> ##                                dat=partialData, dr=dr, n.fft=n.fft, tol=tol)
> ##                       )[3];
> ## EMtime;
>
> #### Generic optimization.
> ## logLike <- function(rates){
> ##   BDloglikelihood.PO(partialDat=partialData, L=exp(rates[1]), m=exp(rates[2]),
> ##                      nu=beta.immig*exp(rates[1]), n.fft=1024);
> ## }
> ## genericEstimates <- optim(initParamMat, logLike,
> ##                           ##method="L-BFGS-B",
> ##                           ##lower=c(0.0001, 0.0001, .0001), upper=c(100,100,100),
> ##                           control=list(fnscale=-1))
> ## print(genericEstimates <- exp(genericEstimates$par))
> ## print(logLike(log(genericEstimates)))
>
>
> ##Optimal appears to be: c(.4077229, .8642744)
> ## Run starting from the optimal to get the right values setup for the CIs:
> initParamMat <- matrix(c(.41,.86),nrow=1);
> names(initParamMat) <- c("lambdahat","muhat")
> iters <- 1;
> EMtime <- system.time(estimators.hist <-
+                       EM.BD.SC(initParamMat=initParamMat, M=iters, beta.immig=beta.immig,
```

```
+                                dat=partialData, dr=dr, n.fft=n.fft, tol=tol)
+                      )[3];

lambdahat     muhat
     0.41      0.86
[1] "The 1 just finished and the new estimators are"
lambdahat     muhat
0.6049384 0.5786720

> EMtime;

elapsed
 15.638

> estimators.hist

          [,1]     [,2]
[1,] 0.4100000 0.860000
[2,] 0.6049384 0.578672

> Lhat <- estimators.hist[iters+1,1]; Lhat

[1] 0.6049384

> Mhat <- estimators.hist[iters+1,2]; Mhat

[1] 0.578672

> MLEs.FullyObserved;

lambdahat     muhat
0.5557757 0.7021495

> ############ end Run EM
```

Part II

Frequentist Confidence Intervals

We are demonstrating the use of the DOBAD package's capability to form asymptotic confidence intervals of the MLEs from the EM algorithm, on a partially observed linear birth-death markov chain. We estimate the information matrix using the method for partially-observed data from Louis (1982). Note that this requires that the estimates for λ and μ are accurate!

```
> IY.a <- getBDinform.PO(partialData, Lhat=Lhat, Mhat=Mhat,
+                          beta.immig=beta.immig, delta=.001)
> print(IY.a);

          [,1]      [,2]
[1,] 17.163060 -9.724563
[2,] -9.724563 10.152374

> zScr <- 1.96;
> Iinv <- solve(IY.a)
> Ldist <- sqrt(Iinv[1,1])*zScr
> Mdist <- sqrt(Iinv[2,2])*zScr
> CI.L <- c(Lhat-Ldist, Lhat+Ldist);CI.L;

[1] -0.09469219  1.30456905

> CI.M <- c(Mhat-Mdist, Mhat+Mdist);CI.M;

[1] -0.3309948  1.4883389

>
```

References

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B* **44,** 226–233.