Smoothing discrete data (II) – using a hidden Markov model as implemented in the `mhsmm` package

Søren Højsgaard and Jared O'Connell

August 22, 2023

We apply the `mhsmm` package for a simple smoothing task. The data pertain to classification of a cows eating behaviour over time. The true eating status S_t is in the vector `cls0` where '1' denotes not eating and '2' denotes eating. The time resolution is one minute. The observed variables x_t in `cls1` are actually not observations per se but the result of a classification obtained by a neural network (using `nnet()` from the `nnet` package).

See also the vignette "Smoothing discrete data (I)" for an alternative approach to smoothing the data.

```
> load("clsX.RData")
> length(cls0)

[1] 8640

> length(cls1)

[1] 8640

> library(mhsmm)

> plot(cls1[1:2000], type='l', ylim=c(.8,2))
> addStates(cls0[1:2000])
```

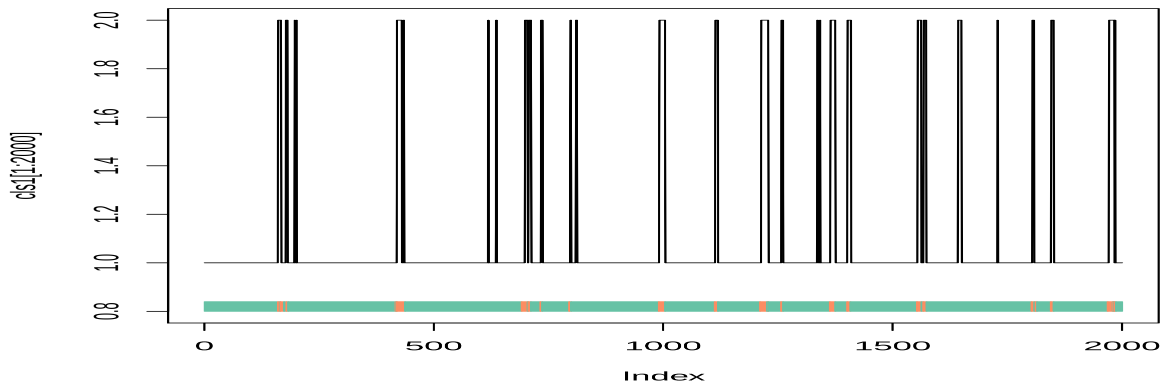

Figure 1: Observed and true eating states

A simple 'smoothing' of the observed states can be obtained as follows:

The density function for the emission distribution is

```
> dpmf <- function(x,j,model) model$parms.emission$pmf[j,x]
```

An initial setting of the parameters is as follows:

```
> J <- 2
> init <- c(.5,.5)
> P <- matrix(c(.9,.1,.1,.9),nrow=J)
> B <- list(pmf=matrix(.1,ncol=J,nrow=J))
> diag(B$pmf) <- .9
> init.spec <- hmmspec(init,trans=P,parms.emission=B,dens.emission=dpmf)
> init.spec

Hidden Markov Model specification:
J (number of states):
2
init:
[1] 0.5 0.5
transition:
     [,1] [,2]
[1,]  0.9  0.1
[2,]  0.1  0.9
emission:
$pmf
     [,1] [,2]
[1,]  0.9  0.1
[2,]  0.1  0.9
```

To fit the model we need to provide the function for the M–step of the EM–algorithm:

```
> mstep.pmf <- function(x,wt) {
+   ans <- matrix(ncol=ncol(wt),nrow=ncol(wt))
+   for(i in 1:ncol(wt))
+     for(j in 1:ncol(wt))
+       ans[i,j] <- sum(wt[which(x==j),i])/sum(wt[,i])
+   list(pmf=ans)
+ }
```

For training the model we use the first 1000 cases

```
> samp <- 1:2640
> train <- list(s=cls0[samp], x=cls1[samp], N=length(cls0[samp]))
> valid <- list(x=cls1[-samp], N=length(cls1[-samp]))
```

We fit the model with

```
> hmm.obj <- hmmfit(train, init.spec,mstep=mstep.pmf)
> summary(hmm.obj)

init:
 1 0

transition:
      [,1]  [,2]
[1,] 0.983 0.017
[2,] 0.177 0.823

emission:
$pmf
            [,1]         [,2]
[1,] 0.999756954 0.0002430464
[2,] 0.008608553 0.9913914471
```

Two types of predictions can be made: Default is to use the Viterbi algorithm for producing the jointly most likely sequence of states given the observed data:

```
> vit <- predict(hmm.obj, valid)
```

Alternatively we can get the individually most likely state sequence as:

```
> smo <- predict(hmm.obj, valid, method="smoothed")
```

The prediction results are quite similar:

```
> normtab <- function(tt) round(sweep(tt,1,rowSums(tt),"/"),2)
> SS <- cls0[-samp]
> XX <- cls1[-samp]
> cls2.vit <- vit$s
> cls2.smo <- smo$s
> normtab(table(SS,XX))

   XX
SS     1    2
  1 0.98 0.02
  2 0.27 0.73

> normtab(table(SS,cls2.vit))

   cls2.vit
SS     1    2
  1 0.98 0.02
  2 0.26 0.74

> normtab(table(SS,cls2.smo))

   cls2.smo
SS     1    2
  1 0.98 0.02
  2 0.26 0.74

> show <- 1:2000
> cls0b <- cls0[-samp]
> cls1b <- cls1[-samp]
> c(length(SS),length(cls2.vit),length(cls2.smo), length(cls0b), length(cls1b))

[1] 6000 6000 6000 6000 6000

> plot(cls1b[show], type='l', ylim=c(.8,2))
> addStates(list(cls0b[show],cls2.vit[show], cls2.smo[show]))
```

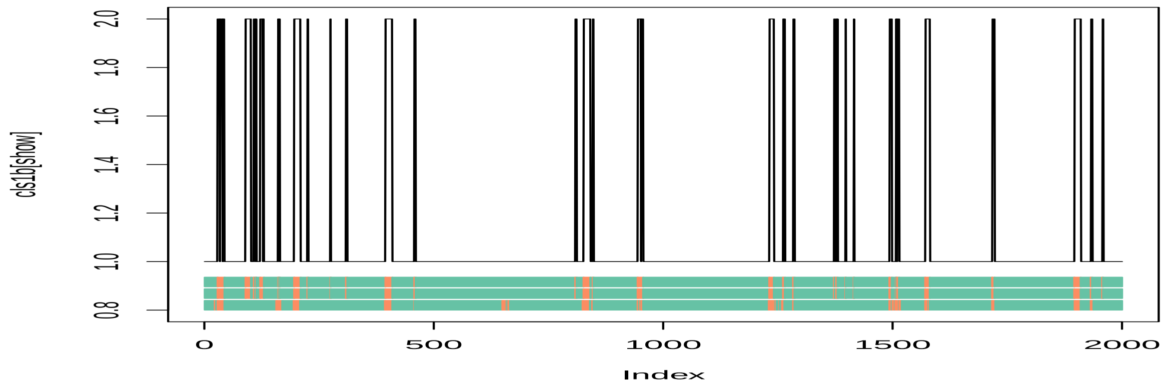

Figure 2: Observed and true eating states