HMM-depmix

Jai Vrat Singh

Let us generate data for training and see if depmix can uncover the pattern

some references: https://eeecon.uibk.ac.at/psychoco/2011/slides/Visser_hdt.pdf

https://quantstrattrader.wordpress.com/2016/10/05/the-problem-with-depmix-with-online-prediction/

https://www.quantstart.com/articles/hidden-markov-models-for-regime-detection-using-r

https://cran.r-project.org/web/packages/depmixS4/depmixS4.pdf

HMM fitting using depmix

Try fitting now..

```
library(depmixS4)
```

```
## Loading required package: nnet
## Loading required package: MASS
## Loading required package: Rsolnp
```

```
hmm <- depmix(obs~1, family = gaussian(), nstates = 2, data = datF)
hmmfit <- fit(hmm)
```

```
## iteration 0 logLik: -2569.649
## iteration 5 logLik: -2538.16
## iteration 10 logLik: -2495.075
## iteration 15 logLik: -2472.191
## iteration 20 logLik: -2456.098
## iteration 25 logLik: -2451.935
## iteration 30 logLik: -2451.653
## iteration 35 logLik: -2451.645
## converged at iteration 38 with logLik: -2451.645
```

```
#the first column has the viterbi states, the other columns have the
# delta probabilities, see Rabiner (1989)
post   <- hmmfit@posterior
post2  <- posterior(hmmfit)
#> head(post)
#  state         S1        S2
#1     2 0.00000000 1.0000000
#2     2 0.08230401 0.9176960

#Must be TRUE as they are same
identical(post, post2)
```

```
## [1] TRUE
```

```
#
head(datF)
```

```
##   state        obs
## 1     2  7.3191971
```

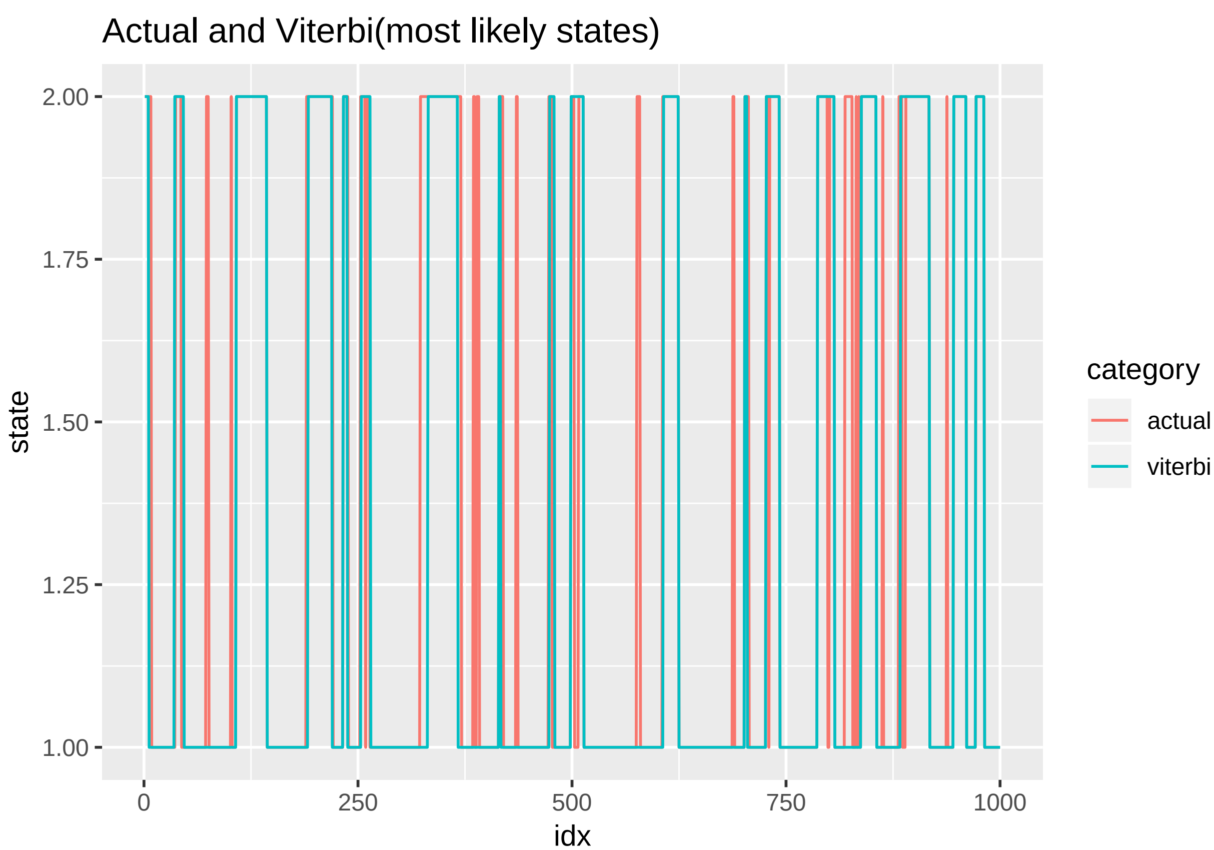
```
## 2      2  8.3811232
## 3      2 -4.1598002
## 4      2  3.9497162
## 5      2  1.9769313
## 6      2  0.7784465
```

```r
library(ggplot2)
library(reshape2)
temp <- data.frame("idx" = 1:dim(datF)[1],"state" = datF$state, "category" = "actual")
head(temp)
```

```
##   idx state category
## 1   1     2   actual
## 2   2     2   actual
## 3   3     2   actual
## 4   4     2   actual
## 5   5     2   actual
## 6   6     2   actual
```

```r
temp <- rbind( temp,
               data.frame("idx" = 1:dim(post)[1], "state" = post$state, "category" = "viterbi"))

# Map  to color
ggplot(data=temp, aes(x=idx, y=state, group=category, colour=category)) +
    geom_line() +
  ggtitle("Actual and Viterbi(most likely states)")
```



Response

```r
hmmfit@response
```

```
## [[1]]
## [[1]][[1]]
## Model of type gaussian (identity), formula: obs ~ 1
## Coefficients:
## (Intercept)
##   -1.042706
## sd  2.056819
##
##
## [[2]]
## [[2]][[1]]
## Model of type gaussian (identity), formula: obs ~ 1
## Coefficients:
## (Intercept)
##    1.747935
## sd  4.145138
```

```r
#mean
hmmfit@response[[1]][[1]]@parameters$coefficients
```

```
## (Intercept)
##   -1.042706
```

```r
#2.047001
#sd
hmmfit@response[[1]][[1]]@parameters$sd
```

```
##       sd
## 2.056819
```

```r
#3.989855

#mean
hmmfit@response[[2]][[1]]@parameters$coefficients
```

```
## (Intercept)
##    1.747935
```

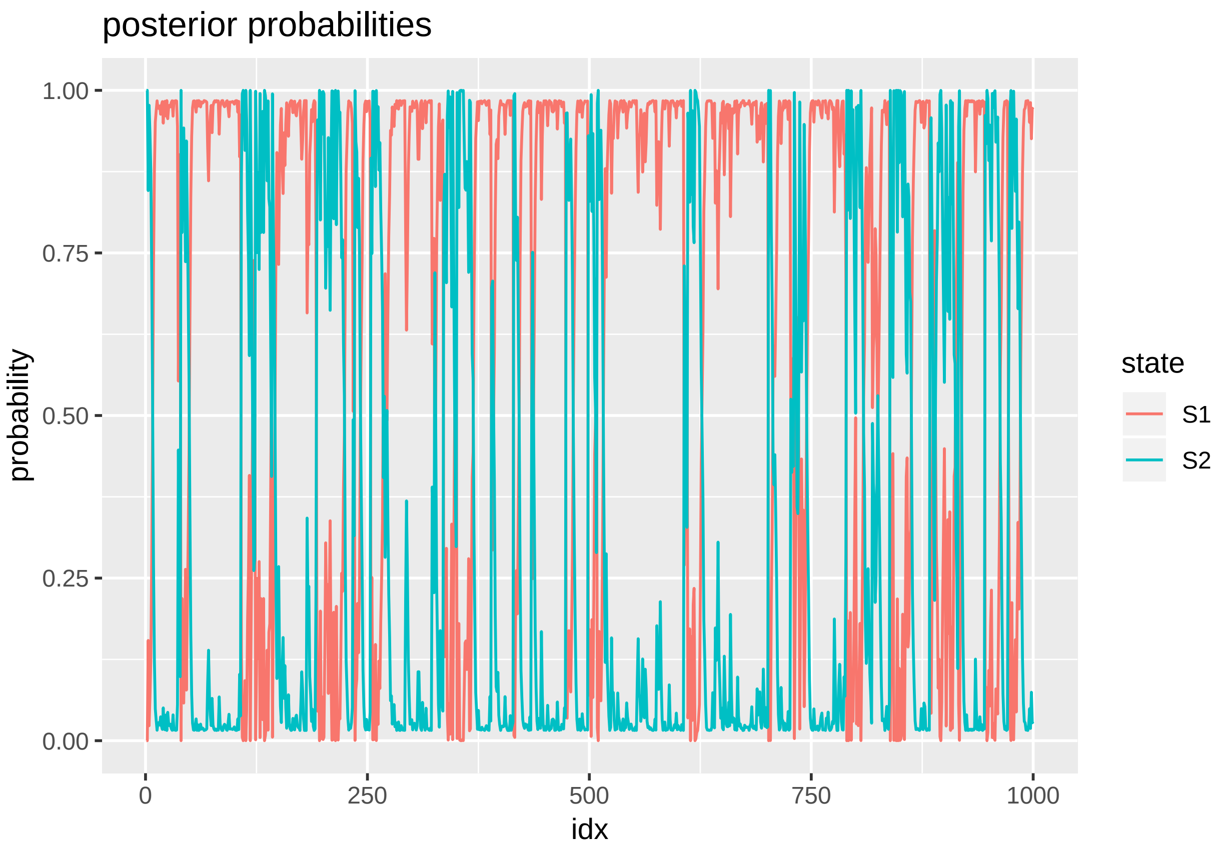
```r
#-0.9999505
#sd
hmmfit@response[[2]][[1]]@parameters$sd
```

```
##       sd
## 4.145138
```

```r
#1.986069
```

We can see that Model is able to uncover the states in graph. Let us plot probabilities.

```r
temp <- data.frame("idx"= 1:dim(post)[1], post[,2:3])
temp.melt <- melt(temp, id.vars = "idx")
colnames(temp.melt) <- c("idx", "state", "probability")
# Map  to color
ggplot(data=temp.melt, aes(x=idx, y=probability, group=state, colour=state)) +
    geom_line() +
  ggtitle("posterior probabilities")
```


posterior probabilities
1.00
0.75
0.50
0.25
0.00
probability
0
250
500
750
1000
idx
state
S1
S2