Hidden Markov model

Aman Kumar Nayak

9/17/2020

Dishonest Casio Sample Case

The dishonest casino gives an example for the application of Hidden Markov Models. This example is taken from Durbin et. al. 1999: A dishonest casino uses two dice, one of them is fair the other is loaded. The probabilities of the fair die are (1/6,...,1/6) for throwing ("1",...,"6"). The probabilities of the loaded die are (1/10,...,1/10,1/2) for throwing ("1",... "5","6"). The observer doesn't know which die is actually taken (the state is hidden), but the sequence of throws (observations) can be used to infer which die (state) was used.

Initiate HMM

```r
library(HMM)

#library(aphid) #for graphical representation

#Setting up intial know parameter

nSim = 2000
States = c("Fair" , "Unfair")
Symbols = 1:6
transProbs = matrix(c(0.99, 0.01, 0.02, 0.98), c(length(States),
     length(States)), byrow = TRUE )

colnames(transProbs) = States
rownames(transProbs) = States

emissionProbs = matrix(c(rep(1/6, 6), c(rep(0.1, 5), 0.5)),
     c(length(States), length(Symbols)), byrow = TRUE)

hmm =HMM::initHMM(States = States, Symbols = Symbols,transProbs =  transProbs,
                  emissionProbs = emissionProbs)

#aphid::plot.HMM(hmm)

#Simulate HMM for given case

SimulateHMM = simHMM(hmm , nSim)

#head(SimulateHMM)
```

Computes the most probable path of states

The Viterbi-algorithm computes the most probable path of states for a sequence of observations for a given Hidden Markov Model.

```
vit = HMM::viterbi(hmm , SimulateHMM$observation)
```

Now computing forward and backward probabilities

The *forward-function* computes the forward probabilities. The forward probability for state X up to observation at time k is defined as the probability of observing the sequence of observations e_1, … ,e_k and that the state at time k is X.

That is: f[X,k] := Prob(E_1 = e_1, … , E_k = e_k , X_k = X).

Where E_1…E_n = e_1…e_n is the sequence of observed emissions and X_k is a random variable that represents the state at time k.

The *backward-function* computes the backward probabilities. The backward probability for state X and observation at time k is defined as the probability of observing the sequence of observations e_k+1, … ,e_n under the condition that the state at time k is X.

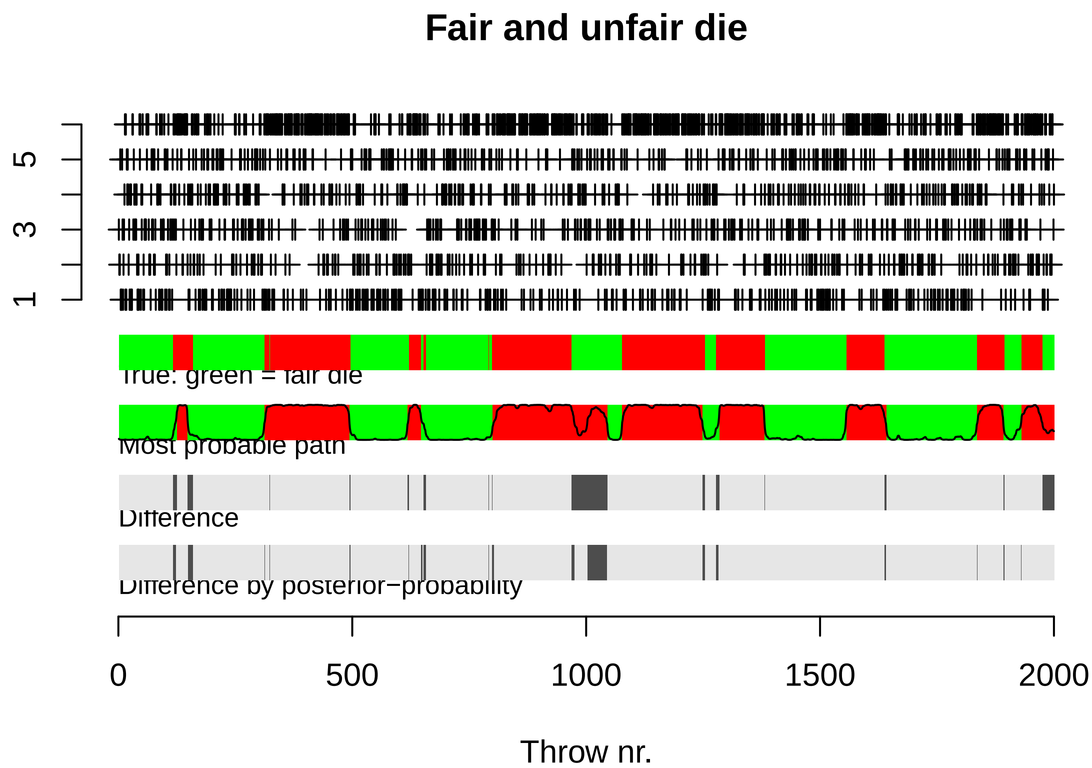
That is: b[X,k] := Prob(E_k+1 = e_k+1, … , E_n = e_n | X_k = X).

Where E_1…E_n = e_1…e_n is the sequence of observed emissions and X_k is a random variable that represents the state at time k.

```
f = HMM::forward(hmm , SimulateHMM$observation)
b = HMM::backward(hmm , SimulateHMM$observation)
```

```
i <- f[1, nSim]
j <- f[2, nSim]
probObservations = (i + log(1 + exp(j - i)))
posterior = exp((f + b) - probObservations)
x = list(hmm = hmm, sim = SimulateHMM, vit = vit, posterior = posterior)

##Plotting simulated throws at top
mn = "Fair and unfair die"
xlb = "Throw nr."
ylb = ""



plot(x$sim$observation, ylim = c(-7.5, 6), pch = 3, main = mn,
    xlab = xlb, ylab = ylb, bty = "n", yaxt = "n")
axis(2, at = 1:6)
#######Simulated, which die was used (truth)###################
 text(0, -1.2, adj = 0, cex = 0.8, col = "black", "True: green = fair die")
 for (i in 1:nSim) {
    if (x$sim$states[i] == "Fair")
        rect(i, -1, i + 1, 0, col = "green", border = NA)
    else rect(i, -1, i + 1, 0, col = "red", border = NA)
   }
########Most probable path (viterbi)#######################
text(0, -3.2, adj = 0, cex = 0.8, col = "black", "Most probable path")
for (i in 1:nSim) {
    if (x$vit[i] == "Fair")
        rect(i, -3, i + 1, -2, col = "green", border = NA)
    else rect(i, -3, i + 1, -2, col = "red", border = NA)
}
##################Differences:
```

```r
text(0, -5.2, adj = 0, cex = 0.8, col = "black", "Difference")
differing = !(x$sim$states == x$vit)
for (i in 1:nSim) {
    if (differing[i])
        rect(i, -5, i + 1, -4, col = rgb(0.3, 0.3, 0.3),
            border = NA)
    else rect(i, -5, i + 1, -4, col = rgb(0.9, 0.9, 0.9),
        border = NA)
       }

 #################Posterior-probability:#########################
 points(x$posterior[2, ] - 3, type = "l")
 ###############Difference with classification by posterior-probability:############
 text(0, -7.2, adj = 0, cex = 0.8, col = "black", "Difference by posterior-probability")
 differing = !(x$sim$states == x$vit)
 for (i in 1:nSim) {
    if (posterior[1, i] > 0.5) {
        if (x$sim$states[i] == "Fair")
            rect(i, -7, i + 1, -6, col = rgb(0.9, 0.9, 0.9),
              border = NA)
        else rect(i, -7, i + 1, -6, col = rgb(0.3, 0.3, 0.3),
            border = NA)
     }
     else {
         if (x$sim$states[i] == "Unfair")
             rect(i, -7, i + 1, -6, col = rgb(0.9, 0.9, 0.9),
               border = NA)
         else rect(i, -7, i + 1, -6, col = rgb(0.3, 0.3, 0.3),
            border = NA)
      }
    }
```



Appendix

```r
knitr::opts_chunk$set(echo = TRUE)

library(HMM)

#library(aphid) #for graphical representation

#Setting up intial know parameter

nSim = 2000
States = c("Fair" , "Unfair")
Symbols = 1:6
transProbs = matrix(c(0.99, 0.01, 0.02, 0.98), c(length(States),
     length(States)), byrow = TRUE )

colnames(transProbs) = States
rownames(transProbs) = States

emissionProbs = matrix(c(rep(1/6, 6), c(rep(0.1, 5), 0.5)),
     c(length(States), length(Symbols)), byrow = TRUE)

hmm =HMM::initHMM(States = States, Symbols = Symbols,transProbs =  transProbs,
                  emissionProbs = emissionProbs)

#aphid::plot.HMM(hmm)
```

```r
#Simulate HMM for given case

SimulateHMM = simHMM(hmm , nSim)

#head(SimulateHMM)


vit = HMM::viterbi(hmm , SimulateHMM$observation)

f = HMM::forward(hmm , SimulateHMM$observation)
b = HMM::backward(hmm , SimulateHMM$observation)

i <- f[1, nSim]
j <- f[2, nSim]
probObservations = (i + log(1 + exp(j - i)))
posterior = exp((f + b) - probObservations)
x = list(hmm = hmm, sim = SimulateHMM, vit = vit, posterior = posterior)

##Plotting simulated throws at top
mn = "Fair and unfair die"
xlb = "Throw nr."
ylb = ""



plot(x$sim$observation, ylim = c(-7.5, 6), pch = 3, main = mn,
    xlab = xlb, ylab = ylb, bty = "n", yaxt = "n")
axis(2, at = 1:6)
#######Simulated, which die was used (truth)####################
 text(0, -1.2, adj = 0, cex = 0.8, col = "black", "True: green = fair die")
 for (i in 1:nSim) {
    if (x$sim$states[i] == "Fair")
        rect(i, -1, i + 1, 0, col = "green", border = NA)
    else rect(i, -1, i + 1, 0, col = "red", border = NA)
   }
########Most probable path (viterbi)#######################
text(0, -3.2, adj = 0, cex = 0.8, col = "black", "Most probable path")
for (i in 1:nSim) {
    if (x$vit[i] == "Fair")
        rect(i, -3, i + 1, -2, col = "green", border = NA)
    else rect(i, -3, i + 1, -2, col = "red", border = NA)
}
##################Differences:
text(0, -5.2, adj = 0, cex = 0.8, col = "black", "Difference")
differing = !(x$sim$states == x$vit)
for (i in 1:nSim) {
    if (differing[i])
        rect(i, -5, i + 1, -4, col = rgb(0.3, 0.3, 0.3),
            border = NA)
    else rect(i, -5, i + 1, -4, col = rgb(0.9, 0.9, 0.9),
        border = NA)
       }
```

```r
################Posterior-probability:########################
points(x$posterior[2, ] - 3, type = "l")
###############Difference with classification by posterior-probability:############
text(0, -7.2, adj = 0, cex = 0.8, col = "black", "Difference by posterior-probability")
differing = !(x$sim$states == x$vit)
for (i in 1:nSim) {
   if (posterior[1, i] > 0.5) {
       if (x$sim$states[i] == "Fair")
           rect(i, -7, i + 1, -6, col = rgb(0.9, 0.9, 0.9),
             border = NA)
       else rect(i, -7, i + 1, -6, col = rgb(0.3, 0.3, 0.3),
           border = NA)
    }
    else {
        if (x$sim$states[i] == "Unfair")
            rect(i, -7, i + 1, -6, col = rgb(0.9, 0.9, 0.9),
              border = NA)
        else rect(i, -7, i + 1, -6, col = rgb(0.3, 0.3, 0.3),
           border = NA)
     }
   }
```