AML lab2

Hyungyum Kim, Milda Poceviciute, Boxi Zhang, Fanny Karelius

2018/9/26

Q1

For the scenario described in the lab instruction, we made the hidden Markov model (HMM):

```
## Loading required package: HMM
```

```
## $States
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $Symbols
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $startProbs
##  1  2  3  4  5  6  7  8  9 10
##  1  0  0  0  0  0  0  0  0  0
##
## $transProbs
##     to
## from   1   2   3   4   5   6   7   8   9  10
##   1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
##   5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   6  0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0
##   7  0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0
##   8  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0
##   9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
##   10 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5
##
## $emissionProbs
##       symbols
## states   1   2   3   4   5   6   7   8   9  10
##     1  0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2
##     2  0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2
##     3  0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0
##     4  0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0
##     5  0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0
##     6  0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##     7  0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0
##     8  0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2
##     9  0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2
##     10 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2
```

Q2

We run the simulation based on our model for 100 times, and we got following results:

```
## $states
##   [1]  1  1  1  1  2  3  3  4  5  5  6  6  7  7  7  7  7  7  7  8  8  8  9
##  [24]  9  9 10 10 10  1  2  2  3  3  4  4  4  5  5  5  6  7  7  8  9 10  1
##  [47]  2  3  3  4  5  5  6  6  7  7  8  8  8  8  9 10 10 10 10  1  1  2  2
##  [70]  2  2  2  3  3  3  4  5  5  5  6  7  8  8  8  8  8  9  9  9 10 10 10
##  [93]  1  1  1  1  1  1  2  3
##
## $observation
##   [1]  2 10  3 10  1  3  5  6  3  3  4  7  9  7  9  6  6  5  8  7 10 10  9
##  [24] 10 10  9  9 10  2 10  2  5  3  2  6  6  4  7  7  6  5  9  7 10 10  3
##  [47]  3  1  3  3  6  5  4  7  7  9  9 10  6  9 10  2  9  9  8  1  3  2  3
##  [70]  4  3  2  5  4  4  2  4  6  4  6  8 10  8  7  6  6  7  8  9 10  1  9
##  [93]  2  2  3  9  2 10  4  1
```

Q3

Function forward in HMM package return log of α, and function backward returns log of β. In order to get the filtering and smoothing probabilities, we use the definitions:

Filtering: $p(Z^t|x^{0:t}) = \frac{\alpha(Z^t)}{\sum_{z^t} \alpha(z^t)}$

Smoothing: $p(Z^t|x^{0:t}) = \frac{\alpha(Z^t)\beta(Z^t)}{\sum_{z^t} \alpha(z^t)\beta(z^t)}$

We also use the Viterbi algortihm implemented in the HMM package to find the most probable path. The results are following(first 5 columns only):

```
## Filtering probabilities:
```

```
##       index
## states 1   2    3    4     5
##     1  1 0.5 0.25 0.25 0.125
##     2  0 0.5 0.50 0.75 0.500
##     3  0 0.0 0.25 0.00 0.375
##     4  0 0.0 0.00 0.00 0.000
##     5  0 0.0 0.00 0.00 0.000
##     6  0 0.0 0.00 0.00 0.000
##     7  0 0.0 0.00 0.00 0.000
##     8  0 0.0 0.00 0.00 0.000
##     9  0 0.0 0.00 0.00 0.000
##     10 0 0.0 0.00 0.00 0.000
```

```
## Smoothing probabilities:
```

```
##       index
## states 1         2         3         4          5
##     1  1 0.7202073 0.4404145 0.1606218 0.03626943
##     2  0 0.2797927 0.5595855 0.8393782 0.49740933
##     3  0 0.0000000 0.0000000 0.0000000 0.46632124
##     4  0 0.0000000 0.0000000 0.0000000 0.00000000
##     5  0 0.0000000 0.0000000 0.0000000 0.00000000
##     6  0 0.0000000 0.0000000 0.0000000 0.00000000
##     7  0 0.0000000 0.0000000 0.0000000 0.00000000
##     8  0 0.0000000 0.0000000 0.0000000 0.00000000
##     9  0 0.0000000 0.0000000 0.0000000 0.00000000
##     10 0 0.0000000 0.0000000 0.0000000 0.00000000
```

```
## Most probable path:
```

```
##   [1]  1  1  1  1  1  2  3  4  4  4  5  6  7  7  7  7  7  7  8  9 10  1  1
##  [24]  1  1  1  1  1  1  1  2  3  3  3  4  4  4  5  5  6  7  8  9 10  1  1
##  [47]  1  1  2  3  4  4  4  5  6  7  7  8  8  8  9 10 10 10 10  1  1  1  1
##  [70]  2  2  2  3  3  3  3  3  4  5  6  7  8  8  8  8  8  9 10  1  1  1  1
##  [93]  1  1  1  1  1  1  2  2
```

Q4

We compute the accuracy of the filtered and smoothed probability distributions, and of the most probable path. Based on the results, we found the smoothed probability distribution showed the highest accuracy.

```
## Accuracy of Filtered probability distribution:
```

```
## [1] 0.56
```

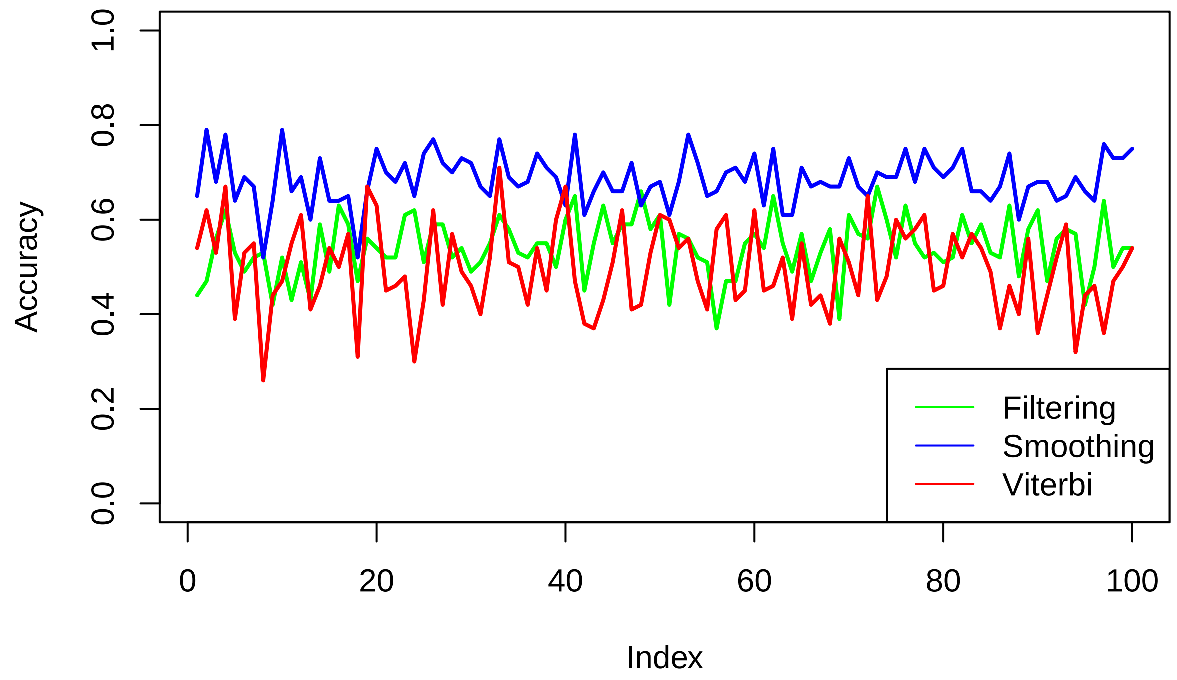
```
## Accuracy of Smoothed probability distribution:
```

```
## [1] 0.75
```

```
## Accuracy of the most probable path:
```

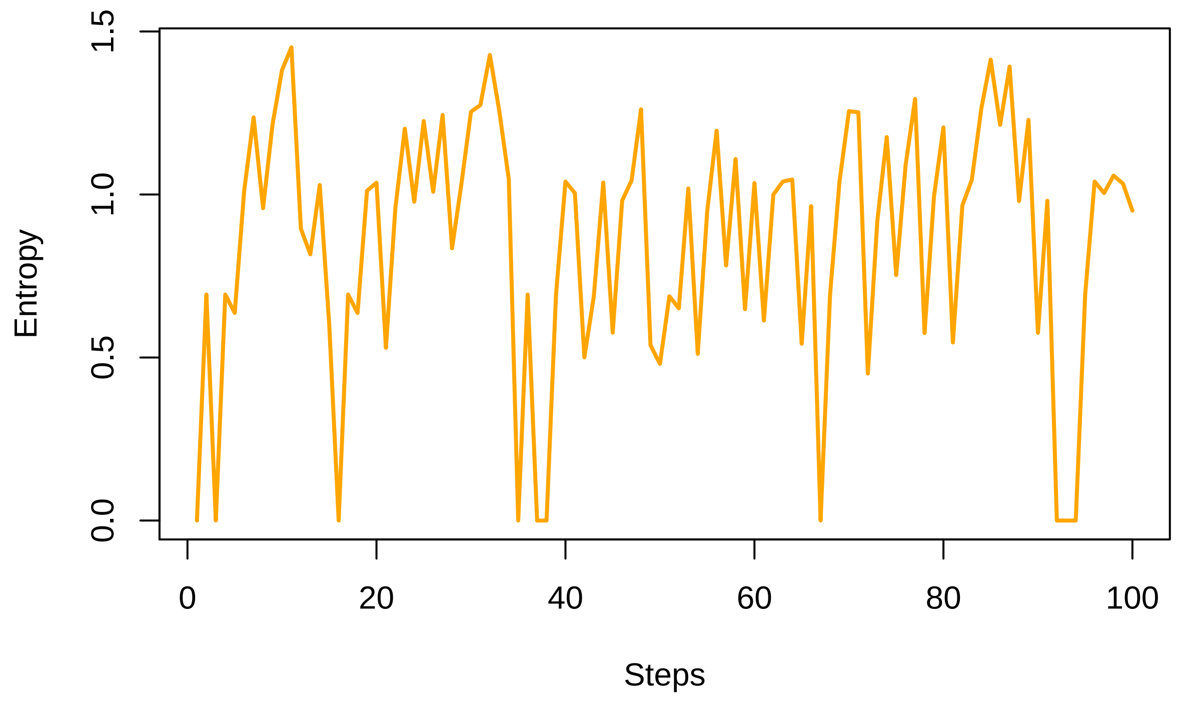
```
## [1] 0.57
```

Q5



From the plot, we see that the accuracy of the Smoothed probability distribution is the highest in almost all cases. This is because, based on the formula, the smoothing includes the information from all observations (so, the observations from time points $0 : t$ as well as from $t + 1 : T$). The filtering includes only the past observations information (that is, from $0 : t$). The Viterbi algortihm finds the most likely plausible path while Smoothing produces the path that has highest probability but it might not be plausible. That's why Smoothing may show better accuracy than the Viterbi algorithm.

Q6



Based on the plot, we see the entropy is fluctuating. We don't see the entropy is decreasing, so there is no evidence that the more observations we have the better we know where the robot is.

Q7

In order to find $p(Z_{101}|X_{0:100})$, we multiply the transition matrix with the filtering probability distribution at the time $t = 100$.

```
##           [,1]       [,2] [,3] [,4] [,5] [,6] [,7] [,8]      [,9]
## [1,] 0.2432432 0.05405405    0    0    0    0    0    0 0.2567568
##          [,10]
## [1,] 0.4459459
```

Hence, it is most likely that the robot would be standing at sector 10.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)

# seed for reproducible results
set.seed(12345)

# package loading
require(HMM)

# Q1.

transP<-diag(x = 0.5, 10, 10, names = TRUE)
for (i in 1:9){
  transP[i,i+1]=0.5
  }
transP[10,1]=0.5

emissionR<-diag(x = 0.2, 10, 10, names = TRUE)
for (i in 1:9){
  emissionR[i,i+1]=0.2
  }
for (i in 1:8){
  emissionR[i,i+2]=0.2
  }
for (i in 2:10){
  emissionR[i,i-1]=0.2
  }
for (i in 3:10){
  emissionR[i,i-2]=0.2
}
  emissionR[1,10]=0.2
  emissionR[2,10]=0.2
  emissionR[1,9]=0.2
  emissionR[10,1]=0.2
  emissionR[10,2]=0.2
  emissionR[9,1]=0.2


fit1 <- initHMM(States = c(1:10),
                Symbols = c(1:10),
                startProbs = c(1,rep(0,9)),
                transProbs = transP,
                emissionProbs = emissionR)

fit1

# Q2.

sim1 <- simHMM(fit1, 100)
sim1
```

```r
# Q3.

filteP <- forward(fit1, sim1$observation)
smoothP <- posterior(fit1, sim1$observation)

filter_probability <- prop.table(exp(filteP),2)
filter_path <- as.vector(apply(filter_probability,2,which.max))

smooth_probability <- prop.table(smoothP,2)
smooth_path <- as.vector(apply(smooth_probability,2,which.max))

Most_Prob_path <- viterbi(fit1, sim1$observation)

cat("Filtering probabilities:")
cat("\n")
filter_probability[,1:5]

cat("Smoothing probabilities:")
cat("\n")
smooth_probability[,1:5]

cat("Most probable path:")
cat("\n")
Most_Prob_path


# Q4.

#get the most probable path
cat("Accuracy of Filtered probability distribution:")
cat("\n")
length(filter_path[filter_path==sim1$states])/length(sim1$states)

cat("Accuracy of Smoothed probability distribution:")
cat("\n")
length(smooth_path[smooth_path==sim1$states])/length(sim1$states)

cat("Accuracy of the most probable path:")
cat("\n")
length(Most_Prob_path[Most_Prob_path==sim1$states])/length(sim1$states)
# Q5.

#initial the vectors for storing the accuracy rates
accuracy_filter <- c()
accuracy_smooth <- c()
accuracy_MPp <- c()

#run 100 iterations for these three methods
for(i in 1:100){

  #simulate from fitted model
  sim2 <- simHMM(fit1, 100)
  #compute the filtered and smoothed probability distributions
  filteP <- forward(fit1, sim2$observation)
  smoothP <- posterior(fit1, sim2$observation)
  Most_Prob_path <- viterbi(fit1, sim2$observation)

  #get the most probable path
  filter_probability <- prop.table(exp(filteP),2)
  filter_path <- as.vector(apply(filter_probability,2,which.max))
  smooth_probability <- prop.table(smoothP,2)
  smooth_path <- as.vector(apply(smooth_probability,2,which.max))

  #compute the percentage of the true hidden states that are guessed by each method.
  accuracy_filter[i] <- length(filter_path[filter_path==sim2$states])/length(sim2$states)
  accuracy_smooth[i] <- length(smooth_path[smooth_path==sim2$states])/length(sim2$states)
  accuracy_MPp[i] <- length(Most_Prob_path[Most_Prob_path==sim2$states])/length(sim2$states)
}

#plot the 100 iterations' accuracies for each
plot(accuracy_filter,type = "l",ylim = c(0,1),col="green",lwd=2,ylab="Accuracy")
lines(accuracy_smooth,col="blue",lwd=2)
lines(accuracy_MPp,col="red",lwd=2)
legend("bottomright",c("Filtering", "Smoothing", "Viterbi"),
col = c("green","blue","red"), lty = c(1, 1, 1))

# Q6.

# Load package
library(entropy)

#experiment on filtered distributions' entropy
#the higher the entropy is, the more uncertainty is.
plot(apply(filter_probability,2,entropy.empirical),type = "l",col="orange",lwd=2,ylab="Entropy",xlab="S

# Q7.
#the posterior of the state at 100 %*% the transition matrix
#get the probabilities of the hidden

filter_probability[,100] %*% transP
most_prob_position <- which.max(filter_probability[,100] %*% transP)
```