AML lab2

Zijie Feng

2019/9/12

Q(1) Build a HMM

```
############ lab2q1 ######################

# state Z: sector the robot might locate
sta <- strsplit("abcdefghij","")[[1]]
# observation x: sector the device reports
sym <- sta
# transition probability matrix
A <- matrix(0, nrow = 10, ncol = 10, dimnames = list(sta,sta))
# possible locations the device reports with different states
B <- matrix(0, nrow = 10, ncol = 10, dimnames = list(sta,sta))
for (i in 1:10) {
  f <- function(i){ifelse(i%%10==0,10,i%%10)}
  A[i,i] <- 0.5
  A[i,f(i+1)] <- 0.5
  B[i,f(i-2)] <- 0.2
  B[i,f(i-1)] <- 0.2
  B[i,f(i)] <- 0.2
  B[i,f(i+1)] <- 0.2
  B[i,f(i+2)] <- 0.2
}
hmm <- initHMM(States = sta, Symbols = sym,
               transProbs = A,
               emissionProbs = B)
```

The transition probability:

	a	b	c	d	e	f	g	h	i	j
a	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
b	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
c	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
d	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0
e	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0
f	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0
g	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0
h	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0
i	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
j	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

The emission probability:

	a	b	c	d	e	f	g	h	i	j
a	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2
b	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2
c	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
d	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0
e	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0
f	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0
g	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0
h	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
i	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2
j	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2

Q(2) Simulate the HMM for 100 time steps

```
########### lab2q2 ######################
set.seed(12345)
n <- 100
res <- simHMM(hmm, n)
states <- res$states
observation <- res$observation
res
```

```
## $states
##   [1] "i" "i" "i" "i" "j" "a" "b" "b" "b" "b" "c" "c" "d" "d" "d" "d" "d"
##  [18] "d" "d" "e" "f" "f" "g" "h" "i" "j" "j" "j" "a" "b" "b" "c" "c" "d"
##  [35] "d" "d" "e" "e" "e" "f" "g" "g" "h" "i" "j" "a" "b" "c" "c" "d" "e"
##  [52] "e" "f" "f" "g" "g" "h" "h" "h" "h" "i" "j" "j" "j" "j" "a" "a" "b"
##  [69] "b" "b" "b" "b" "c" "c" "c" "d" "e" "e" "e" "f" "g" "h" "h" "h" "h"
##  [86] "h" "i" "i" "i" "j" "j" "j" "a" "a" "a" "a" "a" "a" "b" "c"
##
## $observation
##   [1] "g" "j" "h" "j" "b" "c" "j" "c" "d" "d" "e" "d" "b" "c" "b" "f" "f"
##  [18] "e" "d" "c" "e" "e" "h" "i" "j" "i" "i" "j" "b" "j" "b" "e" "c" "b"
##  [35] "f" "f" "d" "g" "g" "f" "e" "i" "g" "j" "j" "c" "c" "a" "c" "c" "f"
##  [52] "e" "d" "g" "g" "i" "i" "j" "f" "i" "j" "b" "i" "i" "h" "a" "c" "b"
##  [69] "c" "d" "c" "b" "e" "d" "d" "b" "d" "f" "d" "f" "h" "j" "h" "g" "f"
##  [86] "f" "g" "h" "i" "j" "a" "i" "b" "b" "c" "i" "b" "j" "d" "a"
```

Q(3) Compute the filtered and smoothed probability distributions and probable paths

```
############ lab2q3 ######################

## the probabilities are in log transformation
# column: probabilities of locations in all sectors
# row: time

fprob <- exp(forward(hmm, observation))
bprob <- exp(backward(hmm, observation))

filtering <- prop.table(fprob,2)              # t(t(fprob)/colSums(fprob))
smoothing <- prop.table(fprob*bprob,2)    #  posterior(hmm, observation)

# get the most possible of path might hav multiple maximumal probabilites
# so we use list to save
fpath <- lapply(1:n, function(i){
  col <- filtering[,i]
  idx <- which.max(col)
  if(sum(col==max(col))>1){
    sta[col==max(col)]
  }else{
    sta[idx]
  }
})

spath <- lapply(1:n, function(i){
  col <- smoothing[,i]
  idx <- which.max(col)
  if(sum(col==max(col))>1){
    sta[col==max(col)]
  }else{
    sta[idx]
  }
})

# viterbi algorithm -- find the path
vpath <- viterbi(hmm, observation)
#  h,i,j,a,a,a...
```

Q(4) Compute the accuracy of the filtered and smoothed path

```
########### lab2q4 ######################

# sta: all possible states
get_accuracy <- function(sta, path, states){
  n <- length(states)
  right <- 0
  for(i in 1:n){
    f <- path[,i]
    fmax <- sta[f==max(f)]
    if(states[i]%in%fmax){ # may hav multiple maximumal probabilites
      right <- right + 1
    }
  }
  return(right/n)
}
rate <- data.frame(filtering_accuracy=get_accuracy(sta, filtering, states),
                   smoothing_accuracy=get_accuracy(sta, smoothing, states))
viterbi_accuracy <- prop.table(table(vpath==states))[2]
rate <- cbind(rate, viterbi_accuracy)
```

```
## the accuracy of filtered path is 0.57
```

```
## the accuracy of smoothed path is 0.74
```

```
## the accuracy of viterbi path is 0.56
```

Q(5) Repeat the previous exercise with different simulated samples

The mean accuracies of filtering and smoothing paths with 100 observations and 50 iterations are

	Mean
filtering_accuracy	0.5825490
smoothing_accuracy	0.6864706
viterbi_accuracy	0.5141176

The general accuracy of smoothing probability distribution is higher than the one of filtering's. Filtering is calculated by

$$p(Z_t|x_{0:t}) = \frac{\alpha(Z_t)}{\sum_{Z_t} \alpha(Z_t)} \quad ,$$

which means that the goal is to calculate the conditional probability of state Z_t given by the previous time series $x_{1:t}$, $t \leq 100$. On the other hand, smoothing is calculated by

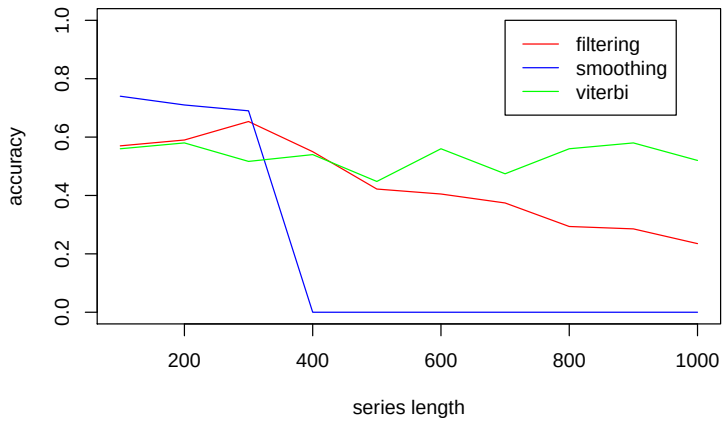
$$p(Z_t|x_{0:t}) = \frac{\alpha(Z_t)\beta(Z_t)}{\sum_{Z_t} \alpha(Z_t)\beta(Z_t)}$$

which is the conditional probability of state Z_t given by the whole time series $x_{1:100}$. This might be the reason why smoothing distribution has such nice preformance.

We also predict the path by Viterbi algorithm for comparison, but the result from smoothed distribution is the highest as well. Another reason is that smoothed distribution predicts each hidden state which is the optimal result given by all the observations. However, it ignores the relation between hidden states and assumes that all time states are independent.

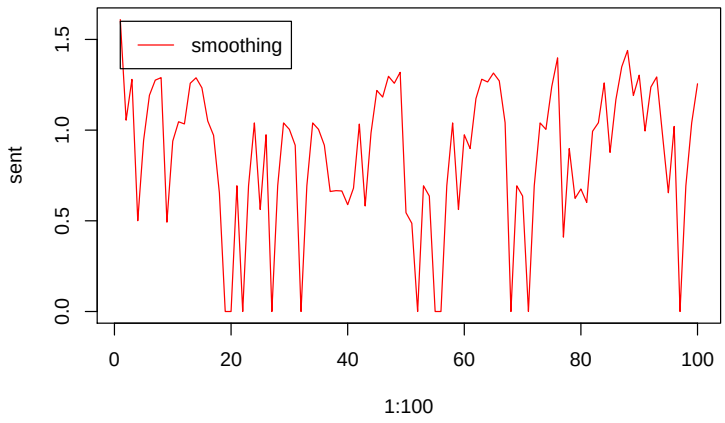
Q(6) Is it true that the more observations you have the better you know where the robot is?

We use a for-loop to calculate the estimation accuracies with different series' lengths.



Besides Viterbi algorithm, the accuracies of filtering and smoothing distributions decrease with the growth of seris' length. Especially for smoothing distribution, both forward and backward algorithms consider quite lot observations which makes the probability of prediction too tiny to calculate. Such bad condition leads smoothed distribution to a zero matrix finally.

Here we use the last simulation (first 100 points) from previous plot with different lengths to calculate their entropies.



Entropy represents the discorder or uncertainty of information, it don't decrease with the increasing number of observations, which confirms that the accuracy cannot increase with the development of observed points.

Q(7) Compute the probabilities of the hidden states for the time step 101.

We still use the simulation from previous plot.

```
########### lab2q7 ######################

# predicition given by x_100 to x_200 by Backward algorithm
bprob <- exp(backward(hmm, observation[100:200]))
col <- prop.table(fprob,2)[,101]
sta[which.max((col))]
```

```
## [1] "f"
```

```
# predicition given by z_100 by FB algorithm
sta[which.max(exp(posterior(hmm, observation))[,100]%*%A )]
```

```
## [1] "g"
```

```
# real
states[101]
```

```
## [1] "g"
```

Appendix

```
########### lab2 ######################
knitr::opts_chunk$set(echo = TRUE, out.height = "200px")
rm(list=ls())
library(kableExtra)
library(HMM)
library(entropy)
########### lab2q1 ######################

# state Z: sector the robot might locate
sta <- strsplit("abcdefghij","")[[1]]
# observation x: sector the device reports
sym <- sta
# transition probability matrix
A <- matrix(0, nrow = 10, ncol = 10, dimnames = list(sta,sta))
# possible locations the device reports with different states
B <- matrix(0, nrow = 10, ncol = 10, dimnames = list(sta,sta))
for (i in 1:10) {
  f <- function(i){ifelse(i%%10==0,10,i%%10)}
  A[i,i] <- 0.5
  A[i,f(i+1)] <- 0.5
  B[i,f(i-2)] <- 0.2
  B[i,f(i-1)] <- 0.2
  B[i,f(i)] <- 0.2
  B[i,f(i+1)] <- 0.2
  B[i,f(i+2)] <- 0.2
}
hmm <- initHMM(States = sta, Symbols = sym,
               transProbs = A,
               emissionProbs = B)
cat("The transition probability:")
kable(A) %>%
  kable_styling(latex_options="basic")
cat("The emission probability:")
kable(B) %>%
  kable_styling(latex_options="basic")
########### lab2q2 ######################
set.seed(12345)
n <- 100
res <- simHMM(hmm, n)
states <- res$states
observation <- res$observation
res
########### lab2q3 ######################

## the probabilities are in log transformation
# column: probabilities of locations in all sectors
# row: time

fprob <- exp(forward(hmm, observation))
bprob <- exp(backward(hmm, observation))

filtering <- prop.table(fprob,2)             # t(t(fprob)/colSums(fprob))
```

```r
smoothing <- prop.table(fprob*bprob,2)    #  posterior(hmm, observation)

# get the most possible of path might hav multiple maximumal probabilites
# so we use list to save
fpath <- lapply(1:n, function(i){
  col <- filtering[,i]
  idx <- which.max(col)
  if(sum(col==max(col))>1){
    sta[col==max(col)]
  }else{
    sta[idx]
  }
})

spath <- lapply(1:n, function(i){
  col <- smoothing[,i]
  idx <- which.max(col)
  if(sum(col==max(col))>1){
    sta[col==max(col)]
  }else{
    sta[idx]
  }
})

# viterbi algorithm -- find the path
vpath <- viterbi(hmm, observation)
#  h,i,j,a,a,a...
############ lab2q4 ########################

# sta: all possible states
get_accuracy <- function(sta, path, states){
  n <- length(states)
  right <- 0
  for(i in 1:n){
    f <- path[,i]
    fmax <- sta[f==max(f)]
    if(states[i]%in%fmax){ # may hav multiple maximumal probabilites
      right <- right + 1
    }
  }
  return(right/n)
}
rate <- data.frame(filtering_accuracy=get_accuracy(sta, filtering, states),
                   smoothing_accuracy=get_accuracy(sta, smoothing, states))
viterbi_accuracy <- prop.table(table(vpath==states))[2]
rate <- cbind(rate, viterbi_accuracy)
cat("the accuracy of filtered path is", rate$filtering_accuracy,"\n")
cat("the accuracy of smoothed path is", rate$smoothing_accuracy,"\n")
cat("the accuracy of viterbi path is", rate$viterbi_accuracy,"\n")
############ lab2q5 ########################

## the iteration number cannot be too much, otherwise we
## get 0 in the final iteration.
```

```r
for (i in 1:50) {
  res <- simHMM(hmm, n)
  states <- res$states
  observation <- res$observation

  fprob <- exp(forward(hmm, observation))
  bprob <- exp(backward(hmm, observation))
  filtering <-  prop.table(fprob,2)
  smoothing <- prop.table(fprob*bprob,2)
  new_rate <- data.frame(filtering_accuracy=get_accuracy(sta, filtering, states),
                     smoothing_accuracy=get_accuracy(sta, smoothing, states))

  vpath <- viterbi(hmm, observation)
  viterbi_accuracy <- prop.table(table(vpath==states))[2]

  new_rate <- cbind(new_rate, viterbi_accuracy)
  rate <- rbind(rate, new_rate)
}
general_rate <- as.data.frame(colMeans(rate))
colnames(general_rate) <- "Mean"

kable(general_rate) %>%        # knitr::kable(general_rate)
  kable_styling(latex_options="basic")
########### lab2q6 ######################

set.seed(12345)
df <- as.data.frame(matrix(nrow = 10, ncol = 3))
len <- seq(100,1000,100)

for(i in 1:10){
  N <- len[i]
  res <- simHMM(hmm, N)
  states <- res$states
  observation <- res$observation
  fprob <- exp(forward(hmm, observation))
  bprob <- exp(backward(hmm, observation))
  filtering <-  prop.table(fprob,2)
  smoothing <- prop.table(fprob*bprob,2)
  vpath <- viterbi(hmm, observation)

  new_rate <- data.frame(filtering_accuracy=get_accuracy(sta, filtering, states),
                         smoothing_accuracy=get_accuracy(sta, smoothing, states))
  viterbi_accuracy <- prop.table(table(vpath==states))[2]
  new_rate <- cbind(new_rate, viterbi_accuracy)

  df[i,] <- new_rate
}
plot(len,df[,1], type="l", col="Red", ylim = c(0,1),xlab="series length",ylab="accuracy")
lines(len,df[,2], col="Blue")
lines(len,df[,3], col="Green")
legend(700,1,legend=c("filtering","smoothing","viterbi"), col=c("red","blue","green"),lty=1)
# # to be simple, we ignore multple maximumal probabilites
# set.seed(12345)
```

```r
# ent <- c()
# for(N in seq(10, 200, 10)){
#   fprob <- exp(forward(hmm, observation[1:N ]))
#   filtering <-  prop.table(fprob,2)
#   fpath <- sapply(1:N, function(k){
#     col <- filtering[,k]
#     idx <- sta[which.max(col)]
#     idx
#   })
#   ent <- c(ent, (entropy.empirical(table(fpath))))
# }
#
# plot(seq(10, 200, 10), ent, xlab="seris length", ylab="entropy",
#      type="l", main="Entropies based on the same simulation with diff. lengths")

sprob <- prop.table(exp(forward(hmm, observation[1:100])),2)
sent <- apply(X = sprob, MARGIN = 2, FUN = entropy.empirical)

plot(1:100, sent, type="l",col="Red")
legend(1,1.6,"smoothing",lty=1, col="Red")
########### lab2q7 ######################

# predicition given by x_100 to x_200 by Backward algorithm
bprob <- exp(backward(hmm, observation[100:200]))
col <- prop.table(fprob,2)[,101]
sta[which.max((col))]

# predicition given by z_100 by FB algorithm
sta[which.max(exp(posterior(hmm, observation))[,100]%*%A )]

# real
states[101]
```