Report - HMM tutorial

Soufian SALIM

October 18, 2013

Abstract

Report on the October 9 tutorial course on R's HMM module.

1 Introduction

From the tutorial's PDF:

> We will use from now the HMM package which is available from the CRAN repository available (http://cran.rstudio.com/). It contains the following functions:
>
> - backward
> - baumWelch
> - dishonestCasino
> - forward
> - initHMM
> - posterior
> - simHMM
> - viterbi
> - viterbiTraining
>
> The goal with this exercise is to verify some of the functions that are available, also to let you understand how all the pieces are working together.

2 Functions

2.1 HMMs_source

2.1.1 Description

Builds two HMMs, using initHMM; the states are "s1" and "s2", and the symbols "a", "b", and "c".

2.1.2 Code

```
HMMs_source <- function() {
  states <- c("s1", "s2")
  obs <- c("a", "b", "c")

  P1_source <- c(1, 0)
  A1_source <- matrix(c(0.1,0.9,0.6,0.4), nrow=2, ncol=2, byrow=T)
  B1_source <- matrix(c(0.1,0.3,0.6,0.4,0.2,0.4), nrow=2, ncol=3,
      byrow=T)

  lambda1_source <- initHMM(states, obs, P1_source, A1_source, B1_
      source)

  P2_source <- c(1, 0)
  A2_source <- matrix(c(0.4,0.6,0.8,0.2), nrow=2, ncol=2, byrow=T)
  B2_source <- matrix(c(0.5,0.4,0.1,0.2,0.1,0.7), nrow=2, ncol=3,
      byrow=T)

  lambda2_source <- initHMM(states, obs, P2_source, A2_source, B2_
      source)

  return(list(m1 = lambda1_source, m2 = lambda2_source))
}
```

2.1.3 Results

Call this function, and check the values of source1_2 ¡- HMMs_source()

HMMs_source() correctly returns two HMMs:

```
$m1
$m1$States
[1] "s1" "s2"

$m1$Symbols
[1] "a" "b" "c"

$m1$startProbs
s1 s2
 1  0

$m1$transProbs
     to
from  s1  s2
  s1 0.1 0.9
  s2 0.6 0.4

$m1$emissionProbs
       symbols
states   a   b   c
     s1 0.1 0.3 0.6
     s2 0.4 0.2 0.4


$m2
$m2$States
[1] "s1" "s2"

```

```
$m2$Symbols
[1] "a" "b" "c"

$m2$startProbs
s1 s2
 1  0

$m2$transProbs
     to
from  s1  s2
  s1 0.4 0.6
  s2 0.8 0.2

$m2$emissionProbs
       symbols
states   a   b   c
    s1 0.5 0.4 0.1
    s2 0.2 0.1 0.7
```

2.2 HMM_simu

2.2.1 Description

Returns a list of two elements: obs1 and obs2, each of them being a sequence of symbols produced using simHMM.

2.2.2 Code

```
HMM_simu <- function(hmm1, hmm2, T = 10){
  return(list(obs1 = simHMM(hmm1, T)$observation, obs2 = simHMM(
      hmm2, T)$observation))
}
```

2.2.3 Results

Call this function, and check the values of train1_2 ¡- HMM_simu(source1_2$m1, source1_2$m2, 1000)

The function returns a series of observations. The possible symbols match those of the HMM defined in HMMs_source():

```
$obs1
   [1] "c" "b" "b" "a" "b" "a" "b" "c" "b" "c" "b" "c" "c" "a" "c"
       "a" "a" "b" "c" "b" "a" "c" "a" "c" "c" "c" "a" "a" "a" "b"
       "a" "c" "c" "a" "a" "c" "c" "b" "c" "c" "c" "a" "c" "c" "c"
       "c" "a" "c" "a" "c" "b" "c"
 (...)
 [989] "c" "a" "b" "c" "c" "a" "a" "c" "c" "c" "b" "c"

$obs2
   [1] "a" "c" "a" "a" "c" "a" "a" "b" "a" "a" "a" "a" "a" "c" "c"
       "a" "b" "b" "c" "b" "c" "a" "a" "c" "a" "c" "a" "c" "a" "a"
       "c" "a" "a" "a" "c" "c" "c" "b" "c" "a" "c" "a" "b" "c" "a"
       "c" "b" "c" "b" "c" "a" "b"
 (...)
 [989] "a" "a" "c" "c" "b" "c" "a" "a" "a" "a" "a" "c"
```

2.3 logLikelihood

2.3.1 Description

Applies the forward algorithm and then ends with the computation of the likelihood of the observation sequence.

2.3.2 Code

```
logLikelihood <- function(hmm, obs){
  fwd <- forward(hmm, obs)

  loglike = fwd[1, length(obs)]

  for(i in 2:length(hmm$States)){
    t = fwd[i, length(obs)]

    if (t > - Inf){
      loglike = t + log(1+exp(loglike-t))
    }
  }

  return(loglike)
}
```

2.3.3 Results

Calculate and display the four log-likelihoods P(O1—lambda1), P(O1—lambda2), P(O2—lambda1) and P(O2—lambda2)

```
P(O1|lambda1) =  -51.71073
P(O1|lambda2) =  -60.70799
P(O2|lambda1) =  -51.70681
P(O2|lambda2) =  -57.3821
```

2.4 HMM_new

2.4.1 Description

Initializes and returns two new models m1 and m2 with initHMM.

2.4.2 Code

```
HMM_new <- function(){
  states <- c("s1","s2")
  obs <- c("a","b","c")

  P1 <-c(1,0)
  A1 <- matrix(c(0.8,0.2,0.5,0.5), nrow=2, ncol=2, byrow=T)
  B1 <- matrix(c(0.3,0.4,0.3,0.1,0.8,0.1), nrow=2, ncol=3, byrow=T)

  M1 <- initHMM(states, obs, P1, A1, B1)
  M2 <- M1

  return(list(m1 = M1, m2 = M2))
}
```

2.5 Check_likelihood_BW

2.5.1 Description

Uses the Baum-Welch algorithm and train one HMM, given a sequence of observations, for 10 iterations, and prints the log-likelihood.

2.5.2 Code

```
checkLikelihoodBW <- function(hmm, obs){
  hmm_new <- list(hmm=hmm, difference=0)

  for(i in 1:10){
    hmm_new <- baumWelch(hmm_new$hmm, obs, 1)
    cat("Likelihood after iteration", i, ": ", logLikelihood(hmm_
        new$hmm, obs), "\n")
  }

  return(hmm_new)
}
```

2.5.3 Results

At each iteration, print the log-likelihood, check that it increases for each iteration. If not, something is wrong in your functions.

The log-likelihood correctly increases at each iteration:

```
Likelihood after iteration 1 :    -104.5332
Likelihood after iteration 2 :    -104.5329
Likelihood after iteration 3 :    -104.5326
Likelihood after iteration 4 :    -104.5322
Likelihood after iteration 5 :    -104.5319
Likelihood after iteration 6 :    -104.5317
Likelihood after iteration 7 :    -104.5314
Likelihood after iteration 8 :    -104.5312
Likelihood after iteration 9 :    -104.531
Likelihood after iteration 10 :   -104.5308

(...)
```

2.6 train_BW

2.6.1 Description

Trains two models from their respective training sequence and returns a list of the two HMMs: m1 and m2.

2.6.2 Code

```
train_BW <- function(training, N = 10) {
  hmms <- HMM_new()

  hmm1 <- baumWelch(hmms$m1,
                    training$obs1,
```

```
6                           N)
7
8   hmm2 <- baumWelch(hmms$m2,
9                     training$obs2,
10                    N)
11
12  return (list(m1 = hmm1$hmm,
13               m2 = hmm2$hmm))
14 }
```

2.6.3 Results

Compare the original source models with the corresponding trained versions.

Common values:

```
$States
[1] "s1" "s2"

$Symbols
[1] "a" "b" "c"

$startProbs
s1 s2
 1  0
```

Original source model:

```
$transProbs
     to
from  s1  s2
  s1 0.1 0.9
  s2 0.6 0.4

$emissionProbs
       symbols
states   a   b   c
     s1 0.1 0.3 0.6
     s2 0.4 0.2 0.4
```

Trained model (10 iterations):

```
$transProbs
     to
from         s1        s2
  s1 0.5639968 0.4360032
  s2 0.3743099 0.6256901

$emissionProbs
       symbols
states          a          b          c
     s1 0.1641866 0.08025629 0.75555710
     s2 0.4012511 0.53199452 0.06675439
```

Discuss and explain your results.

Transition and emission matrices have changed after training to adjust better to the observed data. Given the set of observations, the Baum-Welch algorithm adjusted the probabilities at each iteration to converge towards the maximum likelihood estimate of the HMM parameters.

2.7 classify

2.7.1 Description

Displays the confusion matrix and returns the recognition rate, when using source1_2 to generate N test samples of length T of each source and lambda1_2 as classifiers.

2.7.2 Code

```
classify <- function(source1_2, model1_2, T = 10, N = 1) {
  true_pos = false_pos = true_neg = false_neg = 0

  for (i in 1:(N / 2)) {
    test1_2 <- HMM_simu(source1_2$m1, source1_2$m2, T)

    r <- list(
      o1m1 = logLikelihood(model1_2$m1, test1_2$obs1),
      o1m2 = logLikelihood(model1_2$m2, test1_2$obs1),
      o2m1 = logLikelihood(model1_2$m1, test1_2$obs2),
      o2m2 = logLikelihood(model1_2$m2, test1_2$obs2)
    )

    if (r$o1m1 > r$o1m2)
      true_pos = true_pos + 1
    else
      false_neg = false_neg + 1
    if (r$o2m2 > r$o2m1)
      true_neg = true_neg + 1
    else
      false_pos = false_pos + 1
  }

  print(
    matrix(
      c(true_pos, false_neg, false_pos, true_neg),
      ncol = 2,
      nrow = 2,
      byrow = TRUE,
      dimnames = list(
        c("is true", "is false"),
        c("found true", "found false")
      )
    )
  )

  return (true_pos + true_neg) / N
}
```

2.7.3 Results

Confusion matrix:

	Found True	Found False
Is True	316	184
Is False	73	427

```
           found true found false
```

```
2 is true          316          184
3 is false          73          427
```

Recognition rate:

743 recognized out of 1000 (74.3

2.8 Plot_Reco

2.8.1 Description

Computes and plot the curve which displays the evolution of the recognition rate according to the length of the observed test sequences. Scan a range of 10 values up to Tmax.

2.8.2 Code

```
1  Plot_Reco <- function(source1_2, model1_2, Tmax = 100, N = 1) {
2    reco_rate <- NULL
3    samples_size <- 1:10 * (Tmax/ 10)
4
5    for (i in 1:10 * (Tmax/ 10)) {
6      reco_rate <- c(reco_rate, (classify(source1_2,
7                                          model1_2,
8                                          i,
9                                          N)) / N)
10   }
11
12   plot(samples_size, reco_rate, type = "l", col = "blue", xlab = "
         Samples size", ylab = "Recognition rate")
13 }
```

2.8.3 Results

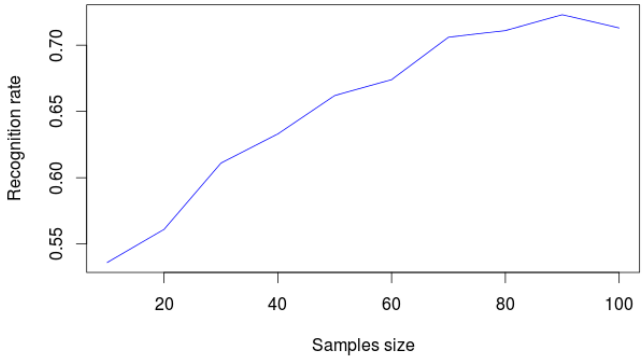

Figure 1: Recognition rate by sample size

2.9 Compare_States

2.9.1 Description

Compares the true sequence of states followed by a model and the one obtained from the Viterbi algorithm.

2.9.2 Code

```
Compare_States <- function(hmm, obs, states) {
  return(
    1 - length(
      Filter(identity, states != viterbi(hmm, obs))
    )
    / length(states)
  )
}
```

2.9.3 Results

Simulate a sequence (states and observations), use the observation to look for the most probable path for this sequence using the source model. What is the percentage of correct assignments??

```
Correct proportion for M1 on O1 compared to Viterbi:  0.51
Correct proportion for M2 on O2 compared to Viterbi:  0.42
```

Would you consider that useful performance?

A result graviting around 50% is hardly useful.

Perform the same experiment but used the trained model instead of source model to do the alignment.

```
Correct proportion for M1 on O1 compared to Viterbi:  0.57
Correct proportion for M2 on O2 compared to Viterbi:  0.6
```

We note a significant improvement in results.

3 Attachments

The code described in this document can be found here: `http://github.com/bolaft/signal_hmm`