Homework 2016-04-18

Chuan Lu
13300180056

April 23, 2016

Problem 1.
Given an HMM with a state transition matrix and display matrix, calculate the probability of the state series HTTT.

Proof. Since the initial condition is not given in this problem, we can suppose that the beginning probability vector is [p, q], where p + q = 1.
Then $f_F(1) = p * 0.5 = 0.5p$,
$f_B(1) = q * 0.8 = 0.8q$,
$f_F(2) = 0.5 * (0.5p * 0.6 + 0.8q * 0.4) = 0.15p + 0.32q$,
$f_B(2) = 0.2 * (0.5p * 0.4 + 0.8q * 0.6) = 0.04p + 0.096q$,
$f_F(3) = 0.5 * (0.6 * (0.15p + 0.32q) + 0.4 * (0.04p + 0.096q)) = 0.053p + 0.1102q$,
$f_B(3) = 0.2 * (0.4 * (0.15p + 0.32q) + 0.6 * (0.04p + 0.096q)) = 0.0168p + 0.03712q$,
$f_F(4) = 0.5 * (0.6 * (0.053p + 0.1102q) + 0.4 * (0.0168p + 0.03712q)) = 0.01926p + 0.040484q$,
$f_B(4) = 0.2 * (0.4 * (0.053p + 0.1102q) + 0.6 * (0.0168p + 0.03712q)) = 0.006256p + 0.0132704q$.
Hence,

$$P(HTTT) = f_F(4) + f_B(4) = 0.025516p + 0.0537544q.$$

□

Problem 2.
Calculate the probability that the hidden state be B for each explicit state.

Proof. According to the basic hypothesis of HMM,
$P(B|H) = \frac{P(B)*P(H|B)}{P(H)} = \frac{(P(B|F)+P(B|B))*P(H|B)}{P(H|B)+P(H|A)} = \frac{0.8}{0.8+0.5} = \frac{8}{13}$,
$P(F|H) = \frac{5}{13}$. □

Problem 3.
Calculate the probability of hidden state series FFBB.

Proof. Like problem 1, suppose the initial probability vector is [p, q], where p + q = 1.
Then according to the basic hypothesis of HMM,
$P(FFBB|HTTT) = \frac{P(FFBB,HTTT)}{P(HTTT)} = \frac{P(F|init)P(H|F)P(F|F)P(T|F)P(B|F)P(T|B)P(B|B)P(T|B)}{P(HTTT)} = \frac{p*0.5*0.6*0.5*0.4*0.2*0.6*0.2}{0.025516p+0.537544q} = \frac{0.00144p}{0.025516p+0.537544q}$. □

Problem 4.
With the assumptions given at problem 1, calculate the most probable hidden route.

Proof. According to Viterbi Algorithm, $v_0(0) = 1, v_B(0) = 0, v_F(0) = 0$;
$v_B(1) = P(x_1 = H|s_1 = B) * max_l P(s_1 = B|init = l)v_l(0) = P(H|B) * q = 0.8q$,
$v_F(1) = P(x_1 = H|s_1 = F) * max_l P(s_1 = F|init = l)v_l(0) = P(H|F) * q = 0.5p$,
$v_B(2) = P(x_2 = T|s_2 = B) * max_l P(s_2 = B|s_1 = l)v_l(1) = max\{0.04p, 0.096q\}$,
$v_F(2) = P(x_2 = T|s_2 = F) * max_l P(s_2 = F|s_1 = l)v_l(1) = max\{0.15p, 0.16q\}$,
$v_B(3) = P(x_3 = T|s_3 = B) * max_l P(s_3 = B|s_2 = l)v_l(2) = max\{0.012p, 0.0128q\}$,

$v_F(3) = P(x_3 = T|s_3 = F) * max_l P(s_3 = F|s_2 = l)v_l(2) = max\{0.045p, 0.048q\}$,
$v_B(4) = P(x_4 = T|s_4 = B) * max_l P(s_4 = B|s_3 = l)v_l(3) = max\{0.0036p, 0.00384q\}$,
$v_F(4) = P(x_4 = T|s_4 = F) * max_l P(s_4 = F|s_3 = l)v_l(3) = max\{0.0135p, 0.0144q\}$.
Hence, The most probable route should be (B if $q \leqslant \frac{5}{13}$ else F), F, F, F. $\square$

Problem 5.
Estimate the state transition matrix and emission matrix with the data given, and estimate the most probable hidden sequence.

Proof. **0.1 The code is shown as follows.**

```
baum_welch = function(data, hidden_states, observed_states, max_iter=1000, max_error=1e-8)

  # init parameters with hidden_states and observed_states;
  transition_matrix = matrix(rep(1, hidden_states^2), ncol = hidden_states)
  # transition_matrix = matrix(runif(hidden_states*hidden_states), nrow = hidden_states)
  transition_matrix = transition_matrix/rowSums(transition_matrix)
  emission_matrix = matrix(runif(hidden_states*observed_states), ncol = observed_states)
  emission_matrix = emission_matrix/rowSums(emission_matrix)

  # instrumental variables, A for transition_matrix, E for emission_matrix
  A = matrix(rep(0, hidden_states*hidden_states), nrow = hidden_states)
  E = matrix(rep(0, hidden_states*observed_states), ncol = observed_states)

  # Number of sequences and length of sequences;
  n_seq = ncol(data)
  len_seq = nrow(data)

  # Iteration
  for (step in 1:max_iter) {

    #print("step")
    print(step)
    # the lists saving the results of forward and backward algorithm, the x_th element
    # is the result of the x_th sequence
    F_list = list()
    B_list = list()
    for (i in 1:n_seq) {
      F_list[[i]] = forward(matrix(data[, i], ncol=1), transition_matrix, emission_matrix)
      B_list[[i]] = backward(matrix(data[, i], ncol=1), transition_matrix, emission_matrix
    }

    # We assume the probabilities for each sequence observed are the same;
    # And each time a multiplication is applied, a const related to the length of the
    # sequence should be multipled to it, so as to enlarge the result.
    # Step 1
    for (k in 1:hidden_states) {
      for (l in 1:hidden_states) {
        temp = 0
        for (j in 1:n_seq) {
          x = data[, j]
          F = F_list[[j]]
          B = B_list[[j]]
          for (i in 1:len_seq) {
```

```
            # Attemtion : We replace the i-1 in F[k ,i] for i
            #print(F[k, i]*transition_matrix[k, l]*emission_matrix[l, x[i]]*B[l, i])
            temp = temp + F[k, i]*transition_matrix[k, l]*emission_matrix[l, x[i]]*B[l, i]
          }
          temp = temp * (len_seq^5)
        }
        A[k, l] = temp
      }
    }

    # Step 2
    for (k in 1:hidden_states) {
      for (b in 1:observed_states) {
        temp = 0
        for (j in 1:n_seq) {
          x = data[, j]
          F = F_list[[j]]
          B = B_list[[j]]
          for (i in 1:len_seq) {
            if (x[i] == b) {
              temp = temp + F[k, i]*B[k, i]
            }
          }
          temp = temp * (len_seq^5)
        }
        E[k, b] = temp
      }
    }

    # Judging of convergence
    #print(A)
    Normalized_A = A/rowSums(A)
    Normalized_E = E/rowSums(E)
    error_A = sum(abs(transition_matrix - Normalized_A))
    error_E = sum(abs(emission_matrix - Normalized_E))
    #print(error_A, error_E)
    if (error_A < max_error & error_E < max_error) {
      break
    }

    # Step 3
    transition_matrix = Normalized_A
    emission_matrix = Normalized_E
    #print(emission_matrix)
  }

  # Return: A list of transition matrix and emission matrix
  returnlist = list(transition_matrix, emission_matrix)
  return(returnlist)
}

forward = function(x, a, e) {
  # forward algorithm;
  # a is the transition matrix and e is the emission matrix;
```

```
  # x is the observed sequence.
  n = nrow(x)

  # hs is the number of hidden states.
  hs = nrow(a)

  # F is the forward matrix;
  # The l_th row and i_th col means f_l(i);
  # The iteration of F is: f_l(i) = e[l, i]*\sigma_k(F[k, i-1] * a[k, l])
  F = matrix(rep(0, n*hs), ncol = n)
  F[, 1] = e[, 1]

  for (i in 2:n) {
    for (l in 1:hs) {
      F[l, i] = e[l, x[i]]*sum(F[, i-1]*a[, l])
    }
  }
  return(F)
}

backward = function(x, a, e) {
  # backward algorithm;
  # x is the observed sequence;
  # a is the transition matrix and e is the emission matrix;
  n = nrow(x)
  hs = nrow(a)

  # B is the backward matrix;
  # the i_th row and j_th col means b_i(j);
  # The iteration of B is: b_i(j) = \sigma_{s_k}P(s_k|s_i)P(x_{j+1}|s_k)b_k(j+1)
  B = matrix(rep(0, n*hs), ncol = n)
  B[, n] = matrix(rep(1, hs), ncol = 1)

  for (j in n-1:-1:1) {
    for (i in 1:hs) {
      B[i, j] = a[i, ] %*% (e[, x[j+1]] * B[, j+1])
    }
  }
  return(B)
}

viterbi = function(x, a, e) {
  # viterbi algorithm
  # x is the observed sequence, a is the state transition matrix,
  # and e is the emission matrix;
  n = nrow(x)
  hs = nrow(a)

  # v is the viterbi matrix;
  # the i_th row and j_th col is v_k(i)
  # The iteration of V is v_k(i) = P(x_i|s_i = k)*max_{l}{P(s_i=k|s_{i-1} = l)}v_l(i-1)
  # Attention : the first column of V is v_k(0)
  v = matrix(rep(0, (n+1)*hs), ncol = n + 1)
  v[1, 1] = 1
```

```
  for (i in 2:(n+1)) {
    for (l in 1:hs) {
      v[l, i] = e[l, x[i-1]] * max(v[, i-1] * a[, l])
    }
  }

  # S is the most probable sequence
  s = matrix(rep(0, n), nrow = 1)
  s[n] = which(v[, n+1] == max(v[, n+1]))

  for (i in (n-1):1) {
    temp = a[, s[i+1]] * v[, i+1]
    s[i] = which(temp == max(temp))
  }
  return(s)
}
```

```
data = read.csv('assign2.csv')
row = nrow(data)
col = ncol(data)
tdata = data[, 2:col]
data = matrix(rep(0, row*(col-1)), nrow = row)
for (i in 1:col-1) {
  d = factor(tdata[, i], levels = c('L', 'R'), labels = c('1', '2'))
  d = as.numeric(d)
  data[, i] = matrix(d)
}

# data = matrix(c(2,2,2,2,2), ncol=1)
source("hmm.R")
hidden_states = 2
observed_states = 2
rlist = baum_welch(data, hidden_states, observed_states)
a = rlist[[1]]
e = rlist[[2]]
s1 = viterbi(matrix(data[, 1]), a, e)
s2 = viterbi(matrix(data[, 2]), a, e)
```

0.2 The result is shown as follows.

```
> a
          [,1]      [,2]
[1,] 0.7871485 0.2128515
[2,] 0.4170624 0.5829376

> e
     [,1] [,2]
[1,]  0.5  0.5
[2,]  0.5  0.5

> s1
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    1    1    1    1    1    1    1    1    1     1     1     1
     [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
```

```
[1,]      1     1     1     1     1     1     1     1     1     1
      [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
[1,]      1     1     1     1     1     1     1      1
> s2
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]     1    1    1    1    1    1    1    1    1     1     1     1
      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92]
[1,]      1     1     1     1     1     1     1     1     1     1
      [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
[1,]      1     1     1     1     1     1     1      1
```

However, every time I rerun the same algorithm, even under the same initial parameters, the result of a, the state transition matrix, can be totally different, while the result of e, the emission matrix, stay all the same. □

Problem 6.

Proof. □