BST249 HW6

Sijia Huo

4/25/2022

1.

```
# initial distribution (m): pi
# transition matrix (m * m): T
# emission matrix (m * k): phi
# observed sequence (n): x

# helper function for log-sum-exp
# take log of elements to sum as input

log_sum_exp = function(log_vec){
  max = max(log_vec)
  log_sum = max + log(sum(exp(log_vec-max)))
  return(log_sum)
}

# forward function
forward = function(log_pi,log_T,log_phi,x){

  # construct matrix to hold result
  n = length(x)
  k = ncol(log_phi)
  m = nrow(log_phi)
  log_s_matrix = matrix(0,n,m) # x by z

  # dynamic programming

  log_s_matrix[1,] = log_pi + log_phi[,x[1]] # for x1

  for(j in 2:n){
    for(i in 1:m){
      # log-sum-exp
      log_s_matrix[j,i] = log_sum_exp(log_s_matrix[j-1,] + log_T[,i] + log_phi[i,x[j]])
    }
  }

  log_px = log_sum_exp(log_s_matrix[n,])
  return(list(mat = log_s_matrix, px = log_px))
}

# backward
backward = function(log_T,log_phi,x){
```

```
  # construct matrix to hold result
  n = length(x)
  k = ncol(log_phi)
  m = nrow(log_phi)
  log_r_matrix = matrix(0,n,m) # x by z

  # dynamic programming

  # when j = n, all entries are 0, no need to change
  for(j in (n-1):1){
    for(i in 1:m){
      # log-sum-exp
      log_r_matrix[j,i] = log_sum_exp(log_T[i,] + log_phi[,x[j+1]] + log_r_matrix[j+1,])
    }
  }
  return(log_r_matrix)
}
```

Check the correctness of the implementation

```
data = read.table("~/Documents/Spring2022/BST249/Homework/HW6/homework-6-data/x.txt",
    quote = "\"", comment.char = "")
data = data[, 1]

# forward & backward
forward_ret = forward(log(rep(1/10, 10)), log(matrix(1/10, 10, 10)), log(matrix(1/59,
    10, 59)), data)
backward_ret = backward(log(matrix(1/10, 10, 10)), log(matrix(1/59, 10, 59)),
    data)

# check correctness
print(forward_ret$px)  # forward p(x)
```

```
## [1] -1190.641
```

```
print(log_sum_exp(log(rep(1/10, 10)) + log(rep(1/59, 10)) + backward_ret[1,
    ]))  # backword p(x)
```

```
## [1] -1190.641
```

Based on our setting, we have equal chance observing any of the $k = 59$ words at each of the $n = 292$ time steps. Therefore, we have $p(x_1, \cdots, x_n) = (\frac{1}{59})^{292}$ and $\log p(x_1, \cdots, x_n) = 292 \log(\frac{1}{59}) = -1190.641$. Therefore, our implementation is correct.

2.

Derive $\varphi_i = (\varphi_{i1}, \ldots, \varphi_{ik})$ where $i = 1 \cdots m$ using Lagrange multipliers as follows: Given $Q(\theta, \theta_k) = \sum_{i=1}^{m} \gamma_{1i} \log \pi_i + \sum_{t=2}^{n} \sum_{i,j=1}^{m} \beta_{tij} \log T_{ij} + \sum_{t=1}^{n} \sum_{i=1}^{m} \gamma_{ti} \log f_{\varphi_i}(x_t)$ and set $f_{\varphi_i}(x_t) = \sum_{K=1}^{k} \mathrm{I}(x_t = K)\varphi_{iK}$, we then have the following equation to estimate φ_{iK} taking account of the constraint that $\sum_{K=1}^{k} \varphi_{iK} = 1$:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial \varphi_{iK}} \left(Q\left(\theta, \theta_k\right) - \lambda \sum_{K=1}^{k} \varphi_{iK} \right) \\
0 &= \frac{\partial}{\partial \varphi_{iK}} \left(\sum_{t=1}^{n} \sum_{i=1}^{m} \gamma_{ti} \log \left(\sum_{K=1}^{k} \mathrm{I}(x_t = K) \varphi_{iK} \right) \right) - \lambda \\
0 &= \frac{\partial}{\partial \varphi_{iK}} \left(\sum_{i=1}^{m} \sum_{K=1}^{k} \sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti} \log\left(\varphi_{iK}\right) \right) - \lambda \\
0 &= \frac{\sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti}}{\varphi_{iK}} - \lambda \\
\varphi_{iK} &= \frac{\sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti}}{\lambda}
\end{aligned}
$$

From the derivation above, we also have $\sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti} = \lambda \varphi_{iK}$. Therefore, we further have $\sum_{K=1}^{k} \sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti} = \sum_{K=1}^{k} \lambda \varphi_{iK} = \lambda \sum_{K=1}^{k} \varphi_{iK} = \lambda$. Since $\sum_{K=1}^{k} \sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti} = \sum_{t=1}^{n} \sum_{K=1}^{k} \mathrm{I}(x_t = K) \gamma_{ti} = \sum_{t=1}^{n} \gamma_{ti}$, we finally have $\lambda = \sum_{t=1}^{n} \gamma_{ti}$ and $\varphi_{iK} = \frac{\sum_{t=1}^{n} \mathrm{I}(x_t = K) \gamma_{ti}}{\sum_{t=1}^{n} \gamma_{ti}}$

Given $\beta_{tij} = \mathbb{P}_{\theta_k}\left(Z_t = i, Z_{t+1} = j \mid x\right)$ (slightly different from the definition on slides) and $p\left(z_j, z_{j+1} \mid x_{1:n}\right) \propto p\left(x_{1:j}, z_j\right) p\left(z_{j+1} \mid z_j\right) p\left(x_{j+1} \mid z_{j+1}\right) p\left(x_{j+2:n} \mid z_{j+1}\right)$, we have

$$
\begin{aligned}
\beta_{tij} &= \frac{p\left(x_{1:t}, z_t = i\right) p\left(z_{t+1} = j \mid z_t = i\right) p\left(x_{t+1} \mid z_{t+1} = j\right) p\left(x_{t+2:n} \mid z_{t+1} = j\right)}{\sum_{i=1}^{m} \sum_{j=1}^{m} p\left(x_{1:t}, z_t = i\right) p\left(z_{t+1} = j \mid z_t = i\right) p\left(x_{t+1} \mid z_{t+1} = j\right) p\left(x_{t+2:n} \mid z_{t+1} = j\right)} \\
&= \frac{s_t\left(z_t = i\right) \cdot T_{ij} \cdot \varphi_{jx_{t+1}} \cdot r_{t+1}\left(z_{t+1} = j\right)}{\sum_{i=1}^{m} \sum_{j=1}^{m} s_t\left(z_t = i\right) \cdot T_{ij} \cdot \varphi_{jx_{t+1}} \cdot r_{t+1}\left(z_{t+1} = j\right)}
\end{aligned}
$$

Given $\gamma_{ti} = \mathbb{P}_{\theta_k}\left(Z_t = i \mid x\right)$ and $p\left(z_j \mid x_{1:n}\right) \propto p\left(x_{1:j}, z_j\right) p\left(x_{j+1:n} \mid z_j\right)$ (from lecture slides), we have

$$
\gamma_{ti} = \frac{p\left(x_{1:t}, z_t = i\right) p\left(x_{t+1:n} \mid z_t = i\right)}{\sum_{j=1}^{m} p\left(x_{1:t}, z_t = i\right) p\left(x_{t+1:n} \mid z_t = i\right)} = \frac{s_t\left(z_t = i\right) r_t\left(z_t = i\right)}{\sum_{i=1}^{m} s_t\left(z_t = i\right) r_t\left(z_t = i\right)} = \sum_{j=1}^{m} \beta_{tij}
$$

With $\pi_i = \frac{\gamma_{1i}}{\sum_{j=1}^{m} \gamma_{1j}}$ and $T_{ij} = \frac{\sum_{t=1}^{n-1} \beta_{tij}}{\sum_{t=1}^{n-1} \sum_{j=1}^{m} \beta_{tij}} = \frac{\sum_{t=1}^{n-1} \beta_{tij}}{\sum_{t=1}^{n-1} \gamma_{ti}}$ along with $p\left(x_{1:n}\right) = \sum_{z_n} s_n\left(z_n\right)$ (from lecture slides), the implementation of Baum–Welch algorithm is as follows:

```
library(gtools)

baum_welch = function(x, m, k) {

    # random initialization
    log_pi = log(rdirichlet(1, rep(1, m)))
    log_T = log(rdirichlet(m, rep(1, m)))
    log_phi = log(rdirichlet(m, rep(1, k)))
    n = length(x)
    forward_ret = forward(log_pi, log_T, log_phi, x)
    log_px_new = forward_ret$px
    converge = FALSE  # flag
    counter = 0  # count the number of ronuds untill convergence

    while (!converge) {
```

```
        counter = counter + 1

        # E step
        log_s = forward_ret$mat
        log_r = backward(log_T, log_phi, x)

        # beta
        log_beta = array(0, dim = c(m, m, n - 1))
        for (t in 1:(n - 1)) {
            for (i in 1:m) {
                # numerator of beta
                log_beta[i, , t] = log_s[t, i] + log_T[i, ] + log_phi[, x[t +
                  1]] + log_r[t + 1, ]
            }
            log_beta_denom = log_sum_exp(log_beta[, , t])
            log_beta[, , t] = log_beta[, , t] - log_beta_denom
        }

        # gamma
        log_gamma = t(apply(log_beta, c(1, 3), log_sum_exp))  # time by state
        log_gamma_tn = apply(log_beta[, , n - 1], 2, log_sum_exp)  # for t=n ;
        log_gamma = rbind(log_gamma, log_gamma_tn)

        # M step
        log_T = apply(log_beta, c(1, 2), log_sum_exp) - apply(log_gamma[1:(n -
            1), ], 2, log_sum_exp)
        for (i in 1:m) {
            log_phi[i, ] = sapply(1:k, function(j) log_sum_exp(log_gamma[which(x ==
                j), i]) - log_sum_exp(log_gamma[, i]))
        }

        log_pi = log_gamma[1, ] - log_sum_exp(log_gamma[1, ])

        # check convergence + E step
        forward_ret = forward(log_pi, log_T, log_phi, x)
        log_px_old = log_px_new
        log_px_new = forward_ret$px
        converge = (abs(log_px_new - log_px_old) < 0.01)
    }

    return(list(log_pi = log_pi, log_T = log_T, log_phi = log_phi, log_px = log_px_new,
        iter = counter))
}
```

Run the algorithm with different m and initializations.

```
bw_ret = lapply(rep(c(10,30,100),each=3), baum_welch, x = data, k = 59)
```

Print out the number of iterations and the $\log p(x_1, \ldots, x_n)$ for each run. For each vector displayed below, the first 3 elements are for $m = 10$, the 4th-6th elements are for $m = 30$ and the last three elements are for $m = 100$.

```
# iterations
sapply(bw_ret, function(ret) ret$iter)
```

```
## [1] 63 39 75 54 83 31 48 35 78
```

```
# log_p_x
sapply(bw_ret, function(ret) ret$log_px)
```

```
## [1] -750.3270 -756.8851 -667.2239 -475.9886 -485.6035 -508.0281 -257.2752
## [8] -270.2100 -240.1957
```

We notice that the number of iterations are not significantly different across different m and the difference is mainly caused by the random initialization. Nevertheless, as the number of hidden states increase, $p(x_1, \ldots, x_n)$ increase. This means that with more hidden states, the overall likelihood at the last step increases and our estimation gets better.

3.

```
# HMM generating function with HMM parameters and sequence length

code <- read.table("~/Documents/Spring2022/BST249/Homework/HW6/homework-6-data/code.txt",
    quote = "\"", comment.char = "")
code_indx = code[, 2]

seq_gen = function(pi, T, phi, N, word_indx) {

    # construct matrix to hold result
    k = ncol(phi)
    m = nrow(phi)

    # generate hidden states
    z = rep(0, N)
    z[1] = sample(1:m, size = 1, prob = pi)  # initial probability
    for (i in 2:N) {
        z[i] = sample(1:m, size = 1, prob = T[z[i - 1], ])
    }

    # generate observed x, map back to word
    x = sapply(1:N, function(i) sample(1:k, size = 1, prob = phi[z[i], ]))
    x_word = sapply(1:N, function(i) word_indx[x[i]])
    x_word = paste(x_word, collapse = " ")
    return(x_word)
}
```

Take the results of the first run for each of $m \in \{10, 30, 100\}$ in question 2 as the estimated parameters, skip step (a). Generate the estimated sequence (step b and c) as follows.

```
# generate and print out sequence
N = 250

# m = 10
ret_10 = bw_ret[[1]]
print(seq_gen(exp(ret_10$log_pi),exp(ret_10$log_T),exp(ret_10$log_phi),N,code_indx))
```

```
## [1] "cute woman climbed brown cute . and with A the sandwich was The tiger ran hat the
table ugly A tiger fish delicious tree . at green banana ugly A fish the tree . The bird
. at a the air was The sandwich sat A big . at A at short . on the tree . little black
ball . The cat . hat at green delicious . a big a in the tiger . under black shirt . bit
```

threw short air . hat in A ate a big and big the shirt . The tiger short the delicious . ate a threw the table ate the shirt ugly in delicious monkey chased a A . cute little had a fat monkey and the tree was and a tall man bird mean a the ball bird The delicious A The woman saw blue The a little woman cute A cat chased the table ugly The monkey ate little dog ate a had a A A a had woman dog ate big sandwich was A blue cute blue little a had little friendly little had woman ran . mean short big bird The bird with the shirt and The dog ran little A a had woman little blue little man had blue little brown yellow delicious cute delicious A monkey in the table . a a brown bit a table . A dog ate A the air . hat the tail big short cat ate a banana . in"

```
# m = 30
ret_30 = bw_ret[[4]]
print(seq_gen(exp(ret_30$log_pi),exp(ret_30$log_T),exp(ret_30$log_phi),N,code_indx))
```

```
## [1] "a little blue bird under the air . The woman with a blue blue banana was bit the
tree . The friendly sat in a short tiger . The tail and a brown yellow cat jumped under
the tail was and a green table . A delicious yellow bird on the tail and sat a big green
table . A monkey bit the big bird was under the air . A cute monkey with a short banana .
A shirt monkey had a blue little woman had a blue ugly hat . The cat bit the banana and
saw the tree was sat a banana was threw a ball chased a little hungry hat . The cat ate
the big ugly dog and a purple black hat was chased the air . A big ugly dog sat a little
hungry hat chased the tree . The cat ran . A friendly orange and saw a tall delicious dog
and a blue hungry bird saw on the sandwich . A little hungry hat and the table . A ugly
bird and a big green tail . A little monkey with a big brown banana . A short sat climbed
a table . The monkey saw the banana . The cat ate the air . The cat ate the tree . The
dog under the table . A big hungry ugly house . The cat bit the tail . A little yellow
hat chased on the tree . The woman in a"
```

```
# m = 100
ret_100 = bw_ret[[7]]
print(seq_gen(exp(ret_100$log_pi),exp(ret_100$log_T),exp(ret_100$log_phi),N,code_indx))
```

```
## [1] "A little cute monkey was in a big sandwich banana . The man ate a delicious
yellow banana was in the tree . The dog bit the table . The woman had a little blue hat .
The dog under the table . A friendly little dog with a brown ugly dog chased the table
and ate the man . A little cute cat in a green blue and chased the tiger . The man had a
friendly little man with a long brown tail and a tall woman with a brown tail and a mean
orange tiger saw a little man sat under a tree and ate the delicious tiger . The man had
a friendly little blue house . The cat under the ball ran . A fat cat in a green cat .
The woman ate the tree . A delicious green was on the table . A mean orange tiger ran . A
fat cat in a hungry cat . The woman chased a big delicious tiger . A short man sat threw
a green ball ran . A cute little man with a brown ugly hat and a tall woman with a long
brown tail . The tiger ran . A friendly little blue hat . The cat bit the table and threw
a ball and the cat and a tall woman with a short tail . The dog chased the banana . A big
hungry black bird was in the air . The dog"
```

Compare the estimated sequence with the original sequence as follows.

```
# original string
paste(sapply(1:292, function(i) code_indx[data[i]]),collapse=" ")
```

```
## [1] "A little cute monkey was in a big green tree . A delicious yellow banana was in
the tree . The little monkey ate the banana . A short man with a brown ugly hat and a
tall woman with a blue shirt sat at a table . The woman had a cute little dog with a
short tail . The man ate a big delicious sandwich . The dog ate the ugly hat under the
table . A big ugly dog chased a little yellow cat . The cat climbed a tree . The dog sat
under the tree . A woman saw the dog under the tree and threw a ball in the air . The dog
```

```
chased the ball and the cat ran . A man had a little blue house . The man had a dog and a
mean hungry cat and a purple fish . The cat ate the delicious fish . A big sandwich was
on the table . The dog climbed the table and ate the sandwich . A short woman had a
friendly monkey with a long brown tail and a little blue hat . The monkey climbed a tall
tree and threw a green banana . The woman chased the banana . A mean orange tiger saw a
little man and chased the man . A little black cat jumped on the tiger . The cat bit the
tiger . The tiger ran . A fat cat in a green shirt sat at a big table . A friendly little
blue bird sat in a tree . A big hungry black bird was in the air . The little bird chased
the big bird and bit the big bird on the tail ."
```

Final comments:

As the number of hidden states increase, our estimation for the latent structure of the text becomes more and more accurate. As a result, the sequence generated using the estimation becomes more and more readable and interpretable. Nevertheless, we still can't restore the original text from our estimation due to the property of the HMM: the hidden state at each time step only depends on the state of the last step.