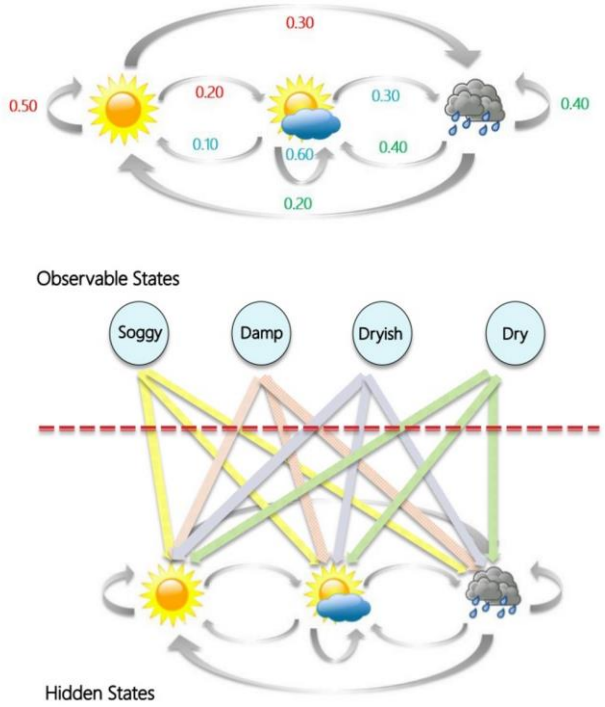
Hidden Markov Models

2016.9.20

장수경

Overview

1. **Introduction**
2. **A multinomial model** of DNA sequence evolution
   - A multinomial model of DNA sequence evolution
   - Generating a DNA sequence using a multinomial model
3. **A Markov model** of DNA sequence evolution
   - A Markov model of DNA sequence evolution
   - The transition matrix for a Markov model
   - Generating a DNA sequence using a Markov model
4. **A Hidden Markov Model** of DNA sequence evolution
   - A Hidden Markov Model of DNA sequence evolution
   - The transition matrix and emission matrix for a HMM
   - Generating a DNA sequence using a HMM
   - Inferring the states of a HMM that generated a DNA sequence
   - A Hidden Markov Model of protein sequence evolution
5. **Summary**



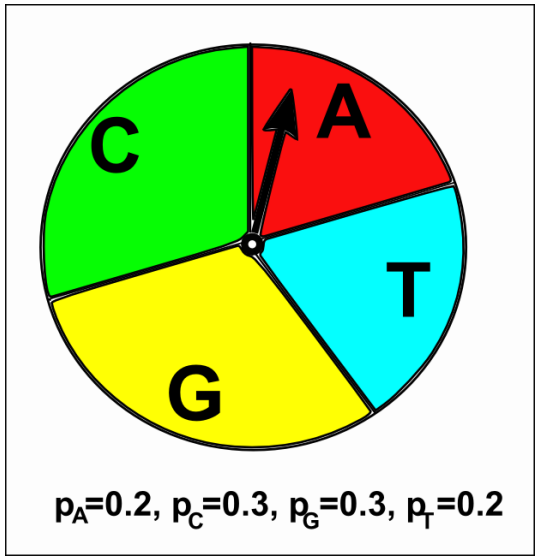
1. Introduction

- **은닉 마르코프 모델(hidden Markov model, HMM)**은 통계적 마르코프 모델의 하나로, 시스템이 은닉된 상태와 관찰가능한 결과의 두 가지 요소로 이루어졌다고 보는 모델이다.

- 은닉 마르코프 모델은 **음성 인식, 필기 인식(Handwriting recognition), 동작 인식(Gesture Recognition), 품사 태깅(Part-of-speech tagging), 악보에서 연주되는 부분을 찾는 작업, 부분 방전(Partial discharge)** 과 같이 시간의 영향을 받는 시스템의 패턴을 인식하는 작업에 유용
  - **생물정보학 분야:** 유전자 예측(Gene prediction), 다중 서열 정렬 (Multiple sequence alignment), DNA 서열 에러 모델링(Modeling DNA seqeuencing error), 단백질 2차 구조 예측(Protein secondary structure prediction), ncRNA 식별(ncRNA identification), 비코딩 RNA의 위치 확인(Fast noncoding RNA annotation)

2.1 A multinomial model of DNA sequence evolution

- **The simplest model of DNA sequence evolution** assumes that the sequence has been produced by **a random process**



2.2 Generating a DNA sequence using a multinomial model

```
> nucleotides    <- c("A", "C", "G", "T") # Define the alphabet of nucleotides
> probabilities1 <- c(0.2, 0.3, 0.3, 0.2) # Set the values of the probabilities
> seqlength      <- 30                    # Set the length of the sequence
> sample(nucleotides, seqlength, rep=TRUE, prob=probabilities1) # Generate a sequence
[1] "A" "C" "T" "G" "T" "T" "T" "T" "A" "G" "T" "C" "A" "G" "G" "G" "G" "C" "G"
[20] "C" "G" "T" "C" "C" "G" "G" "C" "A" "G" "C"
> sample(nucleotides, seqlength, rep=TRUE, prob=probabilities1) # Generate another sequence
[1] "T" "G" "C" "T" "A" "T" "G" "G" "T" "C" "G" "A" "A" "T" "G" "G" "G" "G" "C"
[20] "T" "A" "A" "C" "C" "G" "A" "G" "G" "C" "G"
> probabilities2 <- c(0.1, 0.41, 0.39, 0.1) # Set the values of the probabilities for the new model
> sample(nucleotides, seqlength, rep=TRUE, prob=probabilities2) # Generate a sequence
[1] "G" "C" "C" "T" "C" "C" "C" "C" "G" "G" "G" "G" "G" "A" "C" "C" "C" "A" "G"
[20] "A" "G" "C" "T" "C" "G" "G" "C" "G" "G" "C"
```

sample {base}

Description
sample takes a sample of the specified size from the elements of x using either with or without replacement .

Usage
sample(x, size, replace= FALSE, prob= NULL)

Arguments

x
either a vector of one or more elements from which to choose, or a positive integer.

n
a positive number, the number of items to choose from.

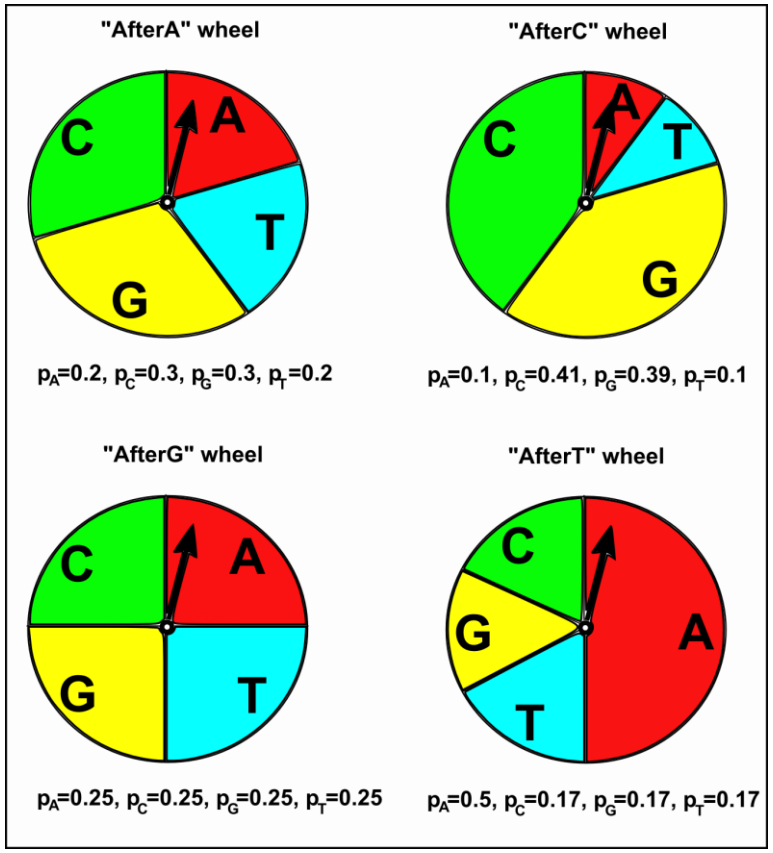
3.1 A Markov model of DNA sequence evolution

- However, for some DNA sequences, **a multinomial model is not an accurate representation** of how the sequences have evolved.

  - One reason is that a multinomial model assumes that **each part of the sequence** (eg. the first 100 nucleotides of the sequence, the second 100 nucleotides, the third 100 nucleotides, etc.) have **the same frequency of each type of nucleotide** (the same pA, pC, pG, and pT)

  - Another assumption of a multinomial model of DNA sequence evolution **does not depend at all on the nucleotides found at adjacent positions** in the sequence. However, for some DNA sequences, This assumption is not true, because the probability of finding a particular nucleotide at a particular position in the sequence *does* depend on what nucleotides are found at adjacent positions in the sequence.

3.1 A Markov model of DNA sequence evolution (cont.)



3.1 A Markov model of DNA sequence evolution (cont.)

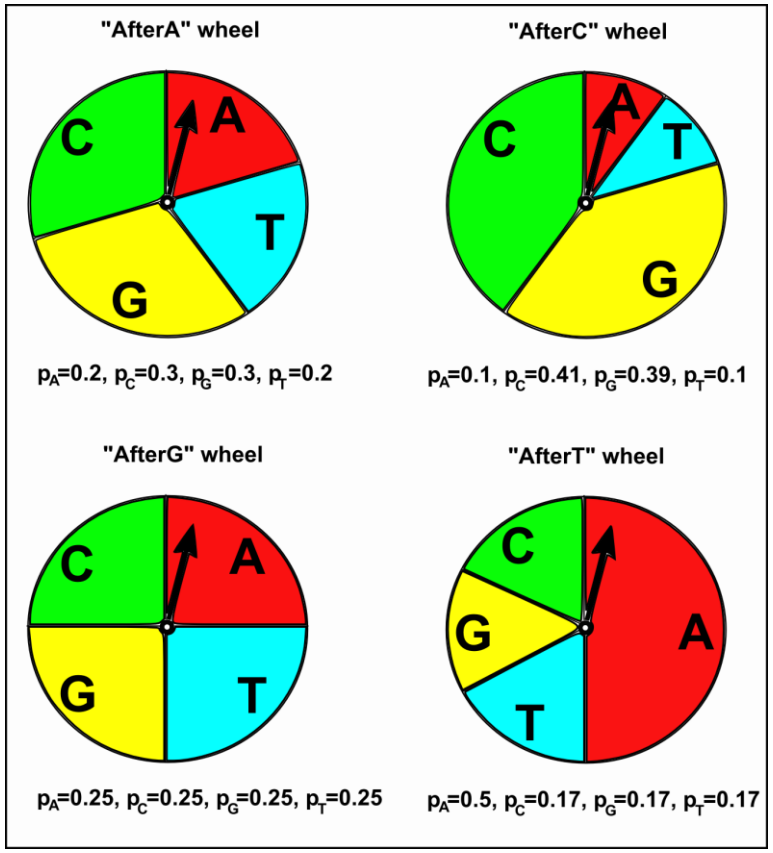
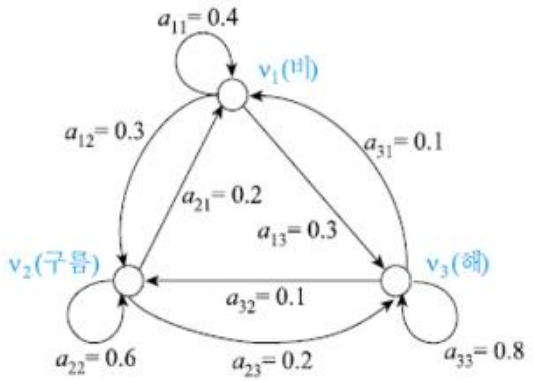


표 7.1 날씨 변화 확률

오늘 \ 내일	비	구름	해
비	0.4	0.3	0.3
구름	0.2	0.6	0.2
해	0.1	0.1	0.8

$$\mathbf{A} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \begin{vmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{vmatrix}$$



(a) 상태 전이 확률 행렬 (b) 상태 전이도

그림 7.4 마크프 모델을 위한 상태 전이 확률과 상태 전이도

3.2 The transition matrix for a Markov model

```
> nucleotides   <- c(" A", "C", "G", "T")      # Define the alphabet of nucleotides
> afterAprobs <- c(0.2, 0.3, 0.3, 0.2)         # Set the values of the probabilities, where the previous nucleotide was "A"
> afterCprobs <- c(0.1, 0.41, 0.39, 0.1)       # Set the values of the probabilities, w here the previous nucleotide was "C"
> after G probs <- c(0.25, 0.25, 0.25, 0.25)     # Set the values of the probabilities, where the previous nucleotide was "G"
> afterTprobs <- c(0.5, 0.17, 0.17, 0.17)      # Set the values of the probabilities, where the previous nucleotide was "T"
> mytransitionmatrix <- matrix(c(after A probs, after C probs, after G probs, after T probs), 4, 4, byrow= TRUE) # Create a 4 x 4 matrix
> rownames(mytransitionmatrix) <- nucleotides
> colnames(mytransitionmatrix) <- nucleotides
> mytransitionmatrix                          # Print out the transition matrix
A    C    G    T
A 0.20 0.30 0.30 0.20
C 0.10 0.41 0.39 0.10
G 0.25 0.25 0.25 0.25
T 0.50 0.17 0.17 0.17
```

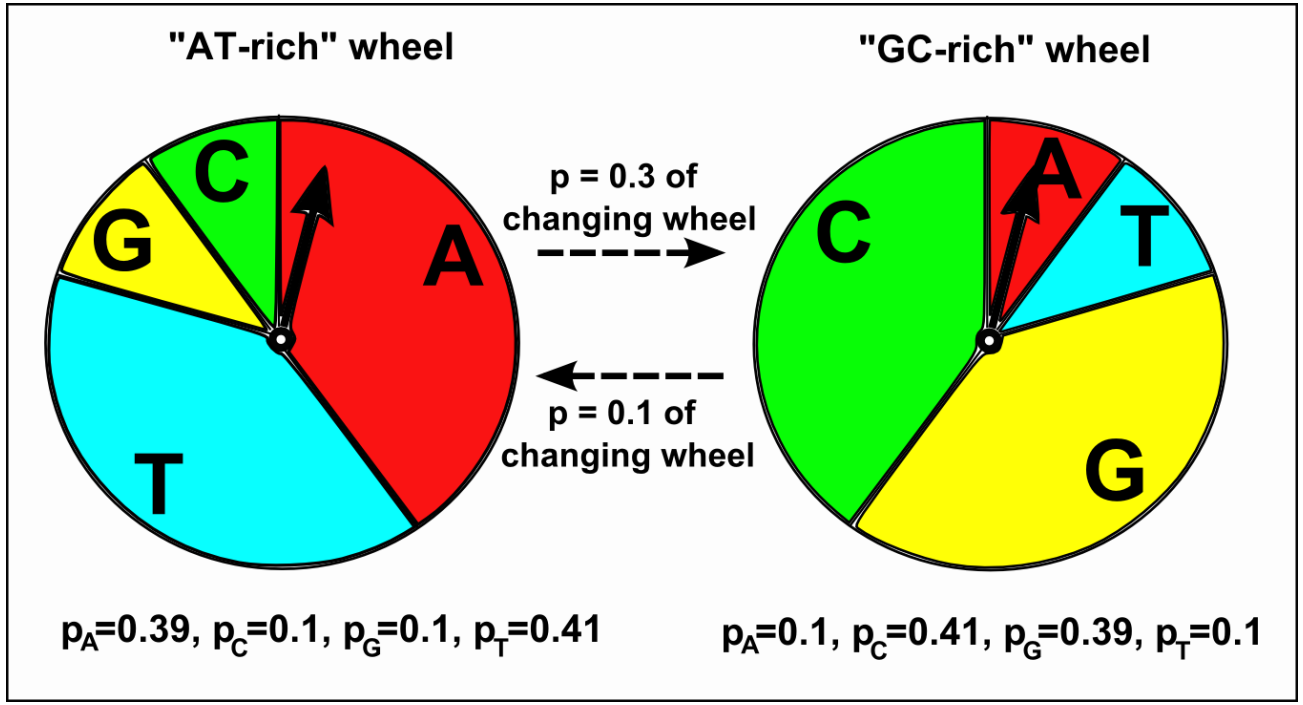
3.3 Generating a DNA sequence using a Markov model

```
> generatemarkovseq <- function(transitionmatrix, initialprobs, seqlength)
{
nucleotides     <- c(" A" , " C", "G", "T") # Define the alphabet of nucleotides
mysequence      <- character()          # Create a vector for storing the ne w sequence
# Choose the nucleotide for the first position in the sequence:
firstnucleotide <- sample(nucleotides, 1, rep=TRUE, prob=initialprobs)
mysequence[1] <- firstnucleotide       # Store the nucleotide for the first position of the sequence
for (i in 2:seqlength)
{
prevnucleotide <- mysequence[i-1]     # Get the previous nucleotide in the new sequence
# Get the probabilities of the current nucleotide, given previous nucleotide "prevnucleotide" :
probabilities  <- transitionmatrix[prevnucleotide,]
# Choose the nucleotide at the current position of the sequence:
nucleotide     <- sample(nucleotides, 1, rep= T RUE, prob=probabilities)
mysequence[i] <- nucleotide          # Store the nucleotide for the current position of the sequence
}
return(mysequence)
}
```

3.3 Generating a DNA sequence using a Markov model

```
> myinitialprobs <- c(0.25, 0.25, 0.25, 0.25)
> generatemarkovseq(mytransitionmatrix, myinitialprobs, 30)
[1] "A" "T" "C" "G" "G" "G" " G" "A" "T" "A" "T" "A" " T" "A" "G" "C" "G" "C" "T" "C" "C"
"C" "G"
[24] "A" "C" "A" "A" "A" "T" "C"
```

4.1 A Hidden Markov Model of DNA sequence evolution



A more complex HMM may model the positions along a sequence as belonging to many different possible states, such as **“promoter”, “exon”, “intron”, and “intergenic DNA”**.

4.1 A Hidden Markov Model of DNA sequence evolution

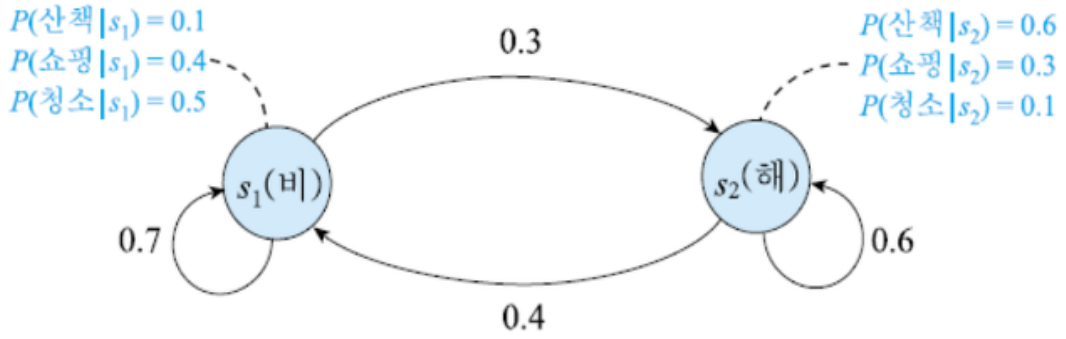


그림 7.9 그녀의 삶을 HMM으로 모델링

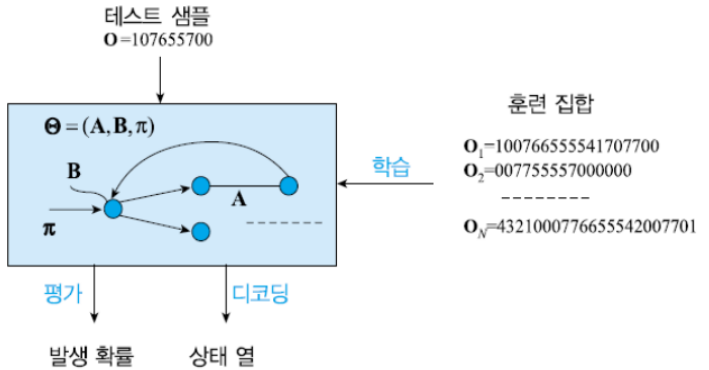


그림 7.11 HMM의 세 가지 문제

답할 수 있는 문제

- 그녀가 일주일 연속으로 쇼핑만 할 확률은 ?

(모델 A가 주어졌을 때 관측 백터 O를 얻었을 때의 확률 P(O|A)는? ; 평가문제) → 동적 프로그래밍 이용

- { 산책, 산책, 청소 } 했다면,
  3일동안 날씨는 각각 어떠했는가 ?

(모델 A가 주어진 상화에서, 관측 벡터 O를 얻었을 때 최적의 상태열은 ? ;디코딩 문제) → 동적 프로그래밍 이용

- { 산책, 산책, 청소} 했다면 오늘과 내일은 무엇을 할까? (복합적인 문제)

(Viterbi **알고리즘**)

훈련집합을 가지고 모델 A를 도출 (학습) → EM 알고리즘 (Baum-Welch 알고리즘)

4.2 The transition matrix and emission matrix for a HMM

```
> states              <- c("AT-rich", "GC-rich") # Define the names of the states
> ATrichprobs         <- c(0.7, 0.3)             # Set the probabilities of switching states, where the previous state was "AT-rich"
> GCrichprobs         <- c(0.1, 0.9)             # Set the probabilities of switching states, where the previous state was "GC-rich"
> thetransitionmatrix <- matrix(c(ATrichprobs, GCrichprobs), 2, 2, byrow = TRUE) # Create a 2 x 2 matrix
> rownames(thetransitionmatrix) <- states
> colnames(thetransitionmatrix) <- states
> thetransitionmatrix                          # Print out the transition matrix
AT-rich GC-rich
AT-rich     0.7     0.3
GC-rich     0.1     0.9
```

4.2 The transition matrix and emission matrix for a HMM

```
> nucleotides        <- c("A", "C", "G", "T")   # Define the alphabet of nucleotides
>ATrichstateprobs    <- c(0.39, 0.1, 0.1, 0.41) # Set the values of the probabilities, for the AT-rich state
>GCrichstateprobs    <- c(0.1, 0.41, 0.39, 0.1) # Set the values of the probabilities, for the GC-rich state
>theemissionmatrix <- matrix(c(ATrichstateprobs, GCrichstateprobs), 2, 4, byrow = TRUE) # Create a 2 x 4 matrix
>rownames(theemissionmatrix) <- states
>colnames(theemissionmatrix) <- nucleotides
> theemissionmatrix                          # Print out the emission matrix
A    C    G    T
AT-rich 0.39 0.10 0.10 0.41
GC-rich 0.10 0.41 0.39 0.10
```

4.3 Generating a DNA sequence using a HMM

```
 # Function to generate a DNA sequence, given a HMM and the length of the sequence to be generated.
generatehmmseq <- function(transitionmatrix, emissionmatrix, initialprobs, seqlength)
  {
     nucleotides      <- c(" A", "C" , " G" , " T")   # Define the alphabet of nucleotides
states           <- c("AT-ri ch" , "GC-rich") # Define the names of the st a tes
mysequence       <- character()            # Create a vector for storing the new sequence
mystates         <- character()           # Create a vector for storing the state that each position in the new sequence
                                          # was generated by
     # Choose the state for the first position in the sequence:
     firststate       <- sample(states, 1, rep= T RUE, prob=initialprobs)
     # Get the probabilities of the current nucleotide, given that we are in the state "firststate" :
probabilities    <- emissionmatrix[firststate,]
     # Choose the nucleotide for the first position in the sequence:
     firstnucleotide <- sample(nucleotides, 1, rep=TRUE, prob=probabilities)
     mysequence[1]    <- firstnucleotide         # Store the nucleotide for the first position of the sequence
     mystates[1]      <- firststate             # Store the state that the first position in the sequence was generated by
```

4.3 Generating a DNA sequence using a HMM

```
for (i in 2:seqlength)
    {
    prevstate     <- mystates[i-1]          # Get the state that the previous nucleotide in the sequence was generated by
      # Get the probabilities of the c urrent state, given that the previous nucleotide was generated by stat e "prevst a te"
stateprobs    <- transitionm a trix[prevst a te,]
      # C hoose the state for the ith position in the sequence:
    state         <- sample(states, 1, rep= T RUE, prob=stateprobs)
      # Get the probabilities of the current nucleotide, given that we are in the state "state":
probabilities <- emissionmatrix[state,]
      # Choose the nucleotide for the ith position in the sequence:
    nucleotide    <- sample(nucleotides, 1, rep=TRUE, prob=probabilities)
    mysequence[i] <- nucleotide             # Store the nucleotide for the current position of the sequence
    mystates[i]   <- state                  # Store the state that the current position in the sequence was generated by
    }
for (i in 1:length(mysequence))
    {
      nucleotide    <- mysequence[i]
      state         <- mystates[i]
      print(paste("Position", i, ", State", state, " , Nucleotide = ", nucleotide))
    }
  }
```

4.3 Generating a DNA sequence using a HMM

```
> theinitialprobs <- c(0.5, 0.5)
> generatehmmseq(thetransitionmatrix, theemissionmatrix, theinitialprobs, 30)
[1] "Position 1 , State AT-rich, Nucleotide=  A"
[1] "Position 2 , State AT-rich, Nucleotide=  A"
[1] "Position 3 , State AT-rich, Nucleotide=  G"
[1] "Position 4 , State AT-rich, Nucleotide=  C"
[1] "Position 5 , State AT-rich, Nucleotide=  G"
[1] "Position 6 , State AT-rich, Nucleotide=  T"
[1] "Position 7 , State GC-rich, Nucleotide =  G"
[1] "Position 8 , St a te GC-rich, Nucleotide=  G"
[1] "Position 9 , St a te GC-rich, Nucleotide=  G"
[1] "Position 10, St a te GC-rich, Nucleotide=  G"
[1] "Position 11, State GC-rich, Nucleotide=  C"
[1] "Position 12, State GC-rich, Nucleotide=  C"
[1] "Position 13, State GC-rich, Nucleotide=  C"
[1] "Position 14, State GC-rich, Nucleotide=  C"
[1] "Position 15, State GC-rich, Nucleotide=  G"
[1] "Position 16, State GC-rich, Nucleotide=  G"
[1] "Position 17, State GC-rich, Nucleotide=  C"
[1] "Position 18, State GC-rich, Nucleotide=  G"
[1] "Position 19, State GC-rich, Nucleotide=  A"
[1] "Position 20, State GC-rich, Nucleotide=  C"
[1] "Position 21, State GC-rich, Nucleotide=  A"
[1] "Position 22, State AT-rich, Nucleotide=  T"
[1] "Position 23, State GC-rich, Nucleotide=  G"
[1] "Position 24, State GC-rich, Nucleotide =  G"
[1] "Position 25, State GC-rich, Nucleotide=  G"
[1] "Position 26, State GC-rich, Nucleotide=  G"
[1] "Position 27, State GC-rich, Nucleotide=  T"
[1] "Position 28, State GC-rich, Nucleotide=  G"
[1] "Position 29, State GC-rich, Nucleotide=  T"
[1] "Position 30, State GC-rich, Nucleotide=  C"
```

4.4 Inferring the states of a HMM that generated a DNA sequence

알고리즘 [7.2] 디코딩을 위한 Viterbi 알고리즘

입력: HMM $\Theta = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, 관측 벡터 $\mathbf{O} = o_1 o_2 \cdots o_T$

출력: 최적 경로 $\hat{\mathbf{Q}} = (\hat{q}_1, \hat{q}_2, \cdots, \hat{q}_T)$

알고리즘:

1. 배열 $\chi[1 \cdots T][1 \cdots n]$와 $\tau[1 \cdots T][1 \cdots n]$을 생성하라.
2. **for** $(i = 1$ **to** $n)$ $\chi[1][i] = \pi_i * b_i(o_1)$; // (7.18)
3. **for** $(t = 2$ **to** $T)$
4. **for** $(i = 1$ **to** $n)$ {
5. $\chi[t-1][j] * a_{ji}$, $1 \le j \le n$ 중에 가장 큰 것의 첨자를 k라 둔다.
6. $\chi[t][i] = \chi[t-1][k] * a_{ki} * b_i(o_t)$; // (7.19)
7. $\tau[t][i] = k$; // (7.19)
8. }
   // 지금부터 (7.20)에 의한 경로 역 추적
9. $\chi[T][j]$, $1 \le j \le n$ 중에 가장 큰 것의 첨자를 $\hat{q}_T$로 한다.
10. **for** $(t = T-1$ **to** $1)$ $\hat{q}_t = \tau[t+1][\hat{q}_{t+1}]$; // (7.20)

4.4 Inferring the states of a HMM that generated a DNA sequence

```
> viterbi <- function(sequence, transitionmatrix, emissionmatrix)
 {
   # Get the names of the states in the HMM:
   states <- rownames(theemissionmatrix)

   # Make the Viterbi matrix v:
   v <- makeViterbimat(sequence, transitionmatrix, emissionmatrix)

   # Go through each of the rows of the matrix v (where each row represents
   # a position in the DNA sequence), and find out which column has the
   # maximum value for that row (where each column represents one state of
   # the HMM):
   mostprobablestatepath <- apply(v, 1, function(x) which.max(x))

   # Print out the most probable state path:
   prevnucleotide <- sequence[1]
   prevmostprobablestate <- mostprobablestatepath[1]
   prevmostprobablestatename <- states[prevmostprobablestate]
   startpos <- 1
```

```
for (i in 2:length(sequence))
   {
      nucleotide <- sequence[i]
      mostprobablestate <- mostprobablestatepath[i]
      mostprobablestatename <- states[mostprobablestate]
      if (mostprobablestatename != prevmostprobablestatename)
      {
         print(paste("Positions",startpos,"-",(i-1), "Most probable state = ",
prevmostprobablestatename))
         startpos <- i
      }
      prevnucleotide <- nucleotide
      prevmostprobablestatename <- mostprobablestatename
   }
   print(paste("Positions",startpos,"-",i, "Most probable state = ",
prevmostprobablestatename))
 }
```

4.4 Inferring the states of a HMM that generated a DNA sequence

```
> makeViterbimat <- function(sequence, transitionmatrix, emissionmatrix)
 {
    # Change the sequence to uppercase
    sequence <- toupper(sequence)
    # Find out how many states are in the HMM
    numstates <- dim(transitionmatrix)[1]
    # Make a matrix with as many rows as positions in the sequence, and as many
    # columns as states in the HMM
    v <- matrix(NA, nrow = length(sequence), ncol = dim(transitionmatrix)[1])
    # Set the values in the first row of matrix v (representing the first position of the sequence) to 0
    v[1, ] <- 0
    # Set the value in the first row of matrix v, first column to 1
    v[1,1] <- 1

 # Fill in the matrix v:
    for (i in 2:length(sequence)) # For each position in the DNA sequence:
    {
       for (l in 1:numstates) # For each of the states of in the HMM:
       {
```

4.4 Inferring the states of a HMM that generated a DNA sequence

```
> mak # Find the probabilility, if we are in state l, of choosing the nucleotide at position in the sequence
 {                statelprobnucleotidei <- emissionmatrix[l,sequence[i]]
   # (
   se(           # v[(i-1),] gives the values of v for the (i-1)th row of v, ie. the (i-1)th position in the sequence.
   # I           # In v[(i-1),] there are values of v at the (i-1)th row of the sequence for each possible state k.
   nu            # v[(i-1),k] gives the value of v at the (i-1)th row of the sequence for a particular state k.
   # I
   # (           # transitionmatrix[l,] gives the values in the lth row of the transition matrix, xx should not be transitionmatrix[,l]?
   v <-          # probabilities of changing from a previous state k to a current state l.
   # S
   v[1           # max(v[(i-1),] * transitionmatrix[l,]) is the maximum probability for the nucleotide observed
   # S           # at the previous position in the sequence in state k, followed by a transition from previous
   v[1           # state k to current state l at the current nucleotide position.

# Fill            # Set the value in matrix v for row i (nucleotide position i), column l (state l) to be:
   for           v[i,l] <-  statelprobnucleotidei * max(v[(i-1),] * transitionmatrix[,l])
   {          }
     }
     return(v)
   }
```

4.4 Inferring the states of a HMM that generated a DNA sequence

```
> myseq <- c("A", "A", "G", "C", "G", "T", "G", "G", "G", "G", "C", "C", "C", "C", "G", "G", "C", "G", "A", "C", "A", "T", "G", "G", "G", "G", "T", "G", "T", "C")

> viterbi(myseq, thetransitionmatrix, theemissionmatrix)

[1] "Positions 1 - 2 Most probable state =  AT-rich"

[1] "Positions 3 - 21 Most probable state =  GC-rich"

[1] "Positions 22 - 22 Most probable state =  AT-rich"

[1] "Positions 23 - 23 Most probable state =  GC-rich"
```

Reference

https://a-little-book-of-r-for-bioinformatics.readthedocs.io/en/latest/src/chapter10.html

https://ko.wikipedia.org/wiki/은닉_마르코프_모델

http://slideplayer.org/slide/5111405/

http://goodtogreate.tistory.com/297