3. Hidden Markov Models - solutions

AUTHOR
EPFL - SV - BIO-463

PUBLISHED
March 5, 2024

Exercise 1: Translate genes, find longest open reading frame

1. Load the file *GeneSequences.fa*
2. Translate the sequences (in 3 frames: only the forward strand)
3. Find the longest ORF in each sequence: use matchPattern for residues "*" and "M"
4. Save them as a fasta file named *orf.fa*

```
library("Biostrings")
genes = readDNAStringSet("GeneSequences.fa")
## translate sequences, then skip 1st and 2nd nucleotide and translate again
##    => 3 frames
frames = translate(c(genes, subseq(genes, 2), subseq(genes, 3)), no.init.codon=
## we will store the longest orf in this vector of type "AAStringSet"
longest.orf = AAStringSet(rep('', length(genes)))
names(longest.orf) = names(genes)

### repeat for each frame
for (nf in 1:length(frames)) {
    frm = frames[[nf]]
    nme = names(frames)[[nf]]
    ### 1. find positions of "stops", namely "*" characters
    stops = matchPattern("*", frm)
    n0 = 1
    ### 2. repeat for each stop found (start = end = position of "*" in the seq
    for (n1 in start(stops)) {
        ### find first start ("M") between n0 and n1
        starts = matchPattern("M", frm[n0:n1])
        if (length(starts) > 0) {
            n0 = n0+start(starts)[1]-1
            if (nchar(longest.orf[nme]) < n1-n0+1) longest.orf[[nme]] = frm[n0:
        }
        ### next search will be between this stop and the next
        n0 = n1
    }
}
longest.orf
```

```
AAStringSet object of length 2:
    width seq                                               names
[1]  1291 MSGPLEGADGGGDPRPGESFCPG...EEEEEGRSSSSPALPTAGNCTS* human_5187
[2]  1237 MDVGKPRGGSSERGASWSPAAAA...CEASVADDSMEKKGNSPLLLQR* Barn_owl_104355733
```

```
writeXStringSet(longest.orf, "orf.fa")
```

Exercise 2: Construct an HMM to find ORFs

- The states are **S1, S2, S3** represent a start codon, **E1, E21, E22, E32, E33** represent the 3 possible stop codons, **B** is background and **In** are "inner" codons.
- The symbols are nucleotides *A, C, G, T*, states other than start and end codons emit uniform probabilities
- The transition probabilities are obvious (all stops have the same probability) except when specified on the schema

```
states = c("B", "S1", "S2", "S3", "I1", "I2", "I3", "E1", "E21", "E22","E31", "
nstates = length(states)
symbols = c("A", "C", "G", "T")
nsym = length(symbols)

### Most emission prob. will be 1/4 so fill the entire matrix with 1/4,
### then modify the different ones
Emat = matrix(1/4, ncol=nsym, nrow=nstates, dimnames=list(states, symbols))
### start is ATG
Emat["S1",] = c(1, 0, 0, 0)
Emat["S2",] = c(0, 0, 0, 1)
Emat["S3",] = c(0, 0, 1, 0)
### stop is one of TAG, TAA, TGA
### first char is A
Emat["E1",] = c(0, 0, 0, 1)
### second is A or G
Emat["E21",] = c(1, 0, 0, 0)
Emat["E22",] = c(0, 0, 1, 0)
### third is A or G after A, and only A after G
Emat["E31",] = c(0, 0, 1, 0)
Emat["E32",] = c(1, 0, 0, 0)

### Transition matrix: according to the schema
Mmat = matrix(0, ncol=nstates, nrow=nstates, dimnames=list(states, states))
Mmat["B", "B"] = 0.98
Mmat["B", "S1"] = 0.02
Mmat["S1", "S2"] = 1
Mmat["S2", "S3"] = 1
Mmat["S3", "I1"] = 1
Mmat["I1", "I2"] = 1
Mmat["I2", "I3"] = 1
Mmat["I3", "I1"] = 0.9
Mmat["E31", "B"] = 0.95
Mmat["E32", "B"] = 0.95
### Sum of probabilities going out of any state must be 1
Mmat["I3", "E1"] = 0.1
Mmat["E31", "S1"] = 0.05
Mmat["E32", "S1"] = 0.05
### These must be calculated so that each path through the E states has probabi
Mmat["E1", "E21"] = 2/3
Mmat["E1", "E22"] = 1/3
Mmat["E21", "E31"] = 0.5
Mmat["E21", "E32"] = 0.5
Mmat["E22", "E32"] = 1
```

- Create the corresponding HMM object
- Plot the HMM schema (see plot.HMM)
- Run the Viterbi algorithm on the segment *1501:1800* of the human gene and display the resulting states

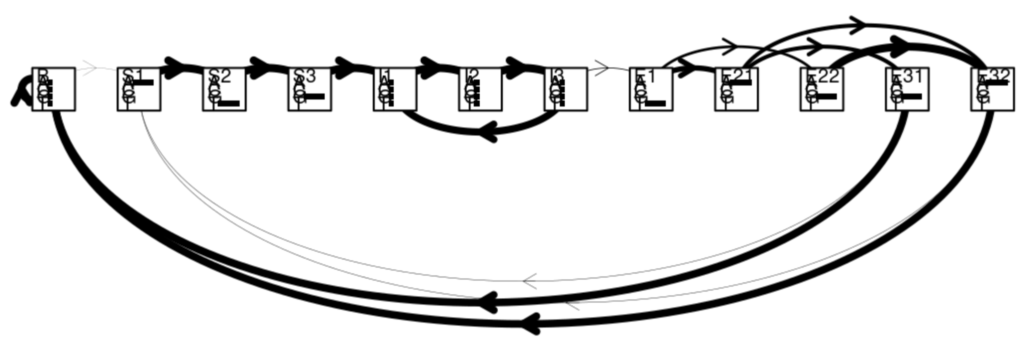
```
library("aphid")

### convert DNA sequence to a list of individual characters,
### keep only the positions 1501-1800
seq = unlist(strsplit(as.character(genes[[1]]), ''))[1501:1800]

### we create an artificial "Begin" that goes directly to "B":
### add 1 row and 1 column to Mmat
M2 = cbind(rep(0, nstates+1), rbind(rep(0, nstates), Mmat))
### label columns and rows with states
dimnames(M2) = list(from=c("Begin", states), to=c("Begin", states))
### transit from "Begin" to "B"
M2["Begin", "B"] = 1
### label rows with states and columns with symbols (ACGT)
dimnames(Emat) = list(from=states, to=symbols)

### Emat / Mmat are in units of probability,
### the HMM calculations are with log(probability)
hmm.orf = structure(list(A=log(M2), E=log(Emat), qe=rep(.25,4)), class="HMM")

plot(hmm.orf)
```




```
hmm.vtrb = Viterbi(hmm.orf, seq)

### for a nice visual display: concatenate all nucleotides into one string
### and show the 1st letter of each state name aligned below
c(paste(seq, collapse=''), paste(substr(states,1,1)[hmm.vtrb$path+1], collapse=
```

```
[1] "GGGGAGTATGTCACCATGGACACCAGCTGGGCTGGCTTTGTGCACCCCTGGAGCCGCAAGGTAGCCTTCGTGTTG
[2] "BBBBBBBSSSIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIEEEBBBBBBBBBBB
```