Series 5

Genomics and bioinformatics - Week 6 - September 23, 2012

This series is about phylogenetic trees and a complement to HMMs.

1 Phylogenetic trees

Circadian rhythms (often referred to as the "body clock") are ubiquitous in most life forms. The "period" family of genes is an essential component of this clock.

The table below provides BLAST scores of pairwise alignments for six period family proteins from three different species,

1. **dm_per** from *Drosophila melanogaster*, i.e. fruit fly,
2. **dr_per2** and **per3** from *Danio rerio*, i.e. zebra fish,
3. **mm_per1**, **mm_per2** and **mm_per3** from *Mus musculus*, i.e. mouse.

	dm_per	dr_per2	dr_per3	mm_per1	mm_per2	mm_per3
dm_per	-	116	125	129	123	121
dr_per2	114	-	661	887	947	497
dr_per3	123	644	-	667	666	699
mm_per1	126	745	572	-	640	80
mm_per2	126	984	680	832	-	455
mm_per3	124	499	709	537	488	-

1. Based on the table above, draw the gene tree for the six period proteins (dm_per, dr_per2, dr_per3, mm_per1, mm_per2, mm_per3).
2. Using this tree, give examples of:
   - an orthologous pair,
   - a paralogous pair in the same species, and
   - a paralogous pair in different species.

2 Another HMM

2.1 The Viterbi algorithm

... fill a 2x4 grid by hand

2.2 Using R

In this exercise we show you how to use an existing package to model a problem with an HMM and solve it. There exist such packages for most programming languages, but here we arbitrarily chose R.

2.2.1 Get started

Start R, then download, install and import the HMM package:

```
install.packages("HMM")
library(HMM)
```

Create an array of hidden states names `states`, an array of emission names `emissions`, a matrix `transProb` with the transition probabilities, a matrix `emissionProb` with the emission probabilities, and a vector `initProb` with the probabilities of the initial state:

```
states <- c("N","I")
emissions <- c("A","T","C","G")
initProb <- c(0.5, 0.5)
transProb <- matrix(c(0.9, 0.45, 0.1, 0.55), 2)
emissionProb <- matrix(c(0.25, 0.05, 0.25, 0.05, 0.25, 0.45, 0.25, 0.45), 2)
```

Create the model with these parameters:

```
hmm <- initHMM(states, emissions, initProb, transProb, emissionProb)
hmm  # prints a summary of the model
```

Using the model created above, let us generate a random observation of length 100:

```
simulation <- simHMM(hmm,100)
simulation  # prints the chosen hidden states and emissions sequences
```

2.2.2 Visualize

Copy and paste the following code to visualize the situation. The plot on top shows the observed sequence, that at the bottom shown the hidden states:

```
obsToVal <- function(obs){
if((obs == "A") or (obs=="T")){return(2)}
else {return(4)}
}
stateToVal <- function(st){
if(st == "I"){return (1)}
if(st == "N"){return (3)}
else {return ("NA")}
}
par(mfrow=c(2,1))
plot( sapply(simulation$observation, obsToVal),pch=19,ylim=c(0,8),col=sapply(simulation$obser
legend("topright",c("A/T","G/C"),pch=19,col=c("red","blue"))
plot( sapply(simulation$states, stateToVal),type="l",ylim=c(0,5))
```

2.2.3 Forward, Backward algorithms

Here are the commands to run the forward and backward algorithms. What is the probability of the observed sequence, given the model? :

```
logForwardProb <- forward(hmm, simulation$observation)
exp(logForwardProb)
logBackwardProbabilities <- backward(hmm, simulation$observations)
exp(logBackwardProb)
```

2.2.4 Viterbi

For the simulated sequence, we knew the hidden states. Now say we have an observed sequences, but do not know the hidden states. Try first to guess where are the "I" region(s) in these two sequences:

```
obs1 <- c("A","T","G","A","A","G","T","A","C","T","T","A","A","G","T","G","A",
          "C","T","A","A","T","T","A","G","T","T","T","A","T","C","T","A","A")
obs2 <- c("A","T","A","G","A","T","T","C","T","G","C","C","G","A","A","A","T",
          "T","T","C","A","T","T","A","G","T","T","T","A","T","C","T","A","A")
```

Now we are going to find the most probable hidden sequence that explains these observations thanks to the Viterbi algorithm. Is it close to what you expected?

```
viterbi1 <- viterbi(hmm, obs1)
viterbi1
viterbi2 <- viterbi(hmm, obs2)
viterbi2
```

2.2.5 Baum-Welsch

Finally, let us admit that our transition and emission probabilities are just a first guess, but we would like to update the model parameters by training it with the Baum-Welsch algorithm:

```
bw = baumWelch(hmm, obs2, 10)  # 10 iterations
print(bw$hmm)
```