Hidden Markov Models - Practical Session 5

Exercise 1 - Fitting a 2-state gamma HMM

The data `elephant.txt` comprises 1000 GPS locations of an African elephant and the according step lengths. We will fit a 2-state gamma HMM to these step lengths.

a) `elephant <- read.table("elephant.txt")`

Familiarise yourself with the data, e.g. by plotting the steps or locations. Would you say that an HMM could be suitable to model the data? Think of possible starting parameters for mean and variance of two gamma distributions for the step lengths (reparametrisation of shape and scale as presented on lecture slide 129).

b) Given the following function, calculate the log-likelihood of the first 52 step lengths and your guessed parameter values for $\theta = (\gamma_{11}, \gamma_{22}, \mu_1, \mu_2, \sigma_1, \sigma_2)$. Use $\gamma_{11} = \gamma_{22} = 0.8$. Then calculate the log-likelihood of the same parameters and the first 100 step lengths. Why do you get an `NA` value and how can you fix that? Assume that missing step length values are missing at random.
Hint: Lecture slides 125, 131

```
llk <- function(theta, x) {
  Gamma <- diag(theta[1:2])
  Gamma[1, 2] <- 1 - Gamma[1, 1]
  Gamma[2, 1] <- 1 - Gamma[2, 2]
  delta <- rep(1 / 2, 2)
  mu <- theta[3:4]
  sigma <- theta[5:6]
  allprobs <- cbind(
    dgamma(x, shape = mu[1]^2 / sigma[1]^2, scale = sigma[1]^2 / mu[1]),
    dgamma(x, shape = mu[2]^2 / sigma[2]^2, scale = sigma[2]^2 / mu[2])
  )
  foo <- delta %*% diag(allprobs[1, ])
  for (t in 2:length(x)) {
    foo <- foo %*% Gamma %*% diag(allprobs[t, ])
  }
  return(log(sum(foo)))
}
```

c) Now that you have adapted the function to account for missing values, try calculating the log-likelihood for the whole 1000 steps. Why do you get an `-Inf` output and how can you scale the forward variables to account for that?
Hint: Lecture slide 117-121

d) The likelihood function is almost ready to use with an optimiser such as `nlm()`! There are two things left to do that `nlm()` cannot handle: 1. We have to account for parameter constraints by defining the likelihood function w.r.t. to an unconstrained parameter vector `theta.star`. 2. We have to get `nlm()` to maximise the log-likelihood instead of minimising it.
Hint: Lecture slide 112-114

e) Run the optimiser and save the results using:

```
mod <- nlm(mllk, theta.star, x = elephant$step, print.level = 2)}
```

Don't forget to transform your starting values and supply `nlm()` with the unconstrained parameter values. Increase the chance of finding the global maximum by running the optimiser with 10 different starting values, e.g. using random sampling from uniform distributions like:

```
theta.star <- c(qlogis(runif(2, 0, 1)), rep(log(runif(2, 1, 400)), 2))
```

If you type `print(k)` in the for-loop over `k in 1:10`, you can track how far the calculations are.
Hint: Lecture slide 123-124

f) Use the results of the model with the highest likelihood. Transform back the parameters of `mod$estimate` and build the matrix $\boldsymbol{\Gamma}$. Additionally, calculate the stationary distribution $\boldsymbol{\delta}$.

g) Visualise the resulting state-depending gamma distributions by plotting their density curves on the histogram of the step lengths. Weight these curves according to the stationary distribution. Additionally, plot the curve of the estimated mixture.