Hidden semi-Markov models

Jan-Ole Koslik

So-called hidden semi-Markov models (HSMMs) are a flexible generalization of HMMs to a semi-Markovian state process which is motivated by the fact that for homogeneous HMMs, the time spent in a hidden state, also called the state dwell time or sojourn time is necessarily geometrically distributed as a consequence of the Markov assumption. HSMMs are designed to mitigate this often unrealistic assumption by allowing for arbitrary distributions on the positive integers to be estimated for the state dwell time. Inference in such models becomes more involved, but Langrock and Zucchini (2011) showed that HSMMs can be estimated conveniently via approximating them by HMMs with an extended state space. Each state of the HSMMs is represented by a state aggregate of several states and the transition probabilities within each aggregate are designed carefully to represent the chosen dwell-time distribution. For more details see Langrock and Zucchini (2011) or Zucchini, MacDonald, and Langrock (2016). Due to this approximate inference procedure, such models can again be fitted by numerically maximizing the (approximate) likelihood which can be evaluated using the forward algorithm.

Homogeneous HSMMs

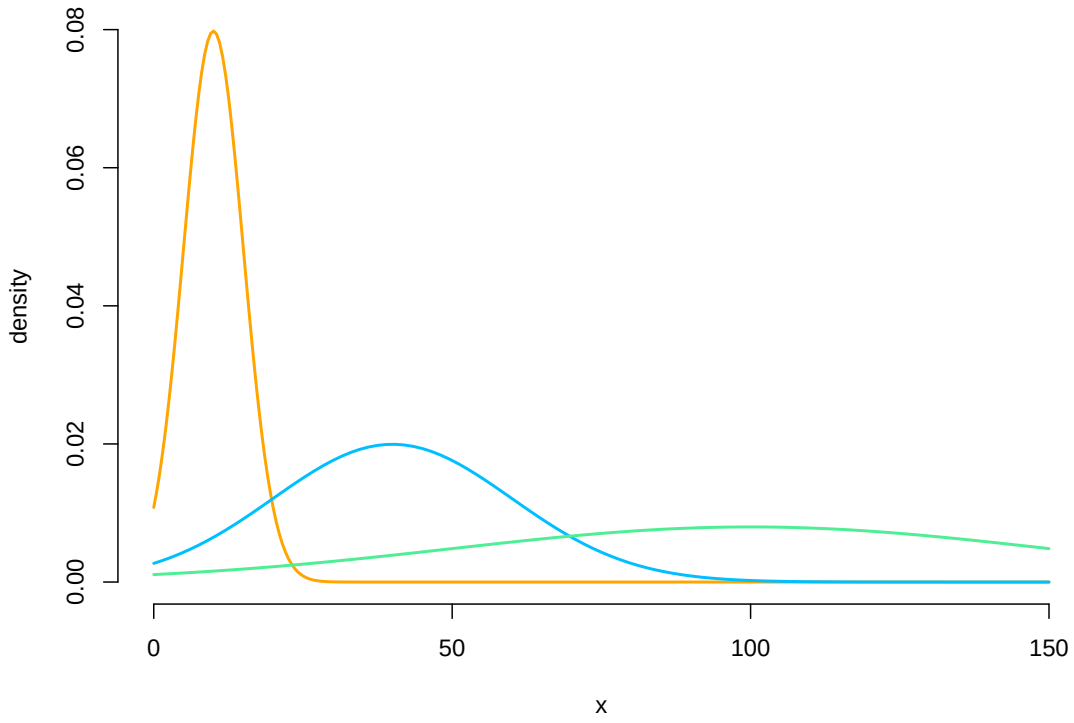
We begin by considering homogeneous HSMMs. In such models, each state has a associated state dwell-time distribution. The transition probability matrix of a regular HMM is the replaced by these distributions and the conditional transition probabilities given the state is left.

Setting parameters

Here we choose the simplest case of dwell times that are **shifted Poisson** distributed. We have to specify the Poisson mean for each state, the conditional transition probability matrix called Ω and the parameters of the state-dependent process.

```
lambda = c(7, 4, 4)
omega = matrix(c(0, 0.7, 0.3,
                 0.5, 0, 0.5,
                 0.7, 0.3, 0), nrow = 3, byrow = TRUE)
mu = c(10, 40, 100)
sigma = c(5, 20, 50)

color = c("orange", "deepskyblue", "seagreen2")
curve(dnorm(x, mu[1], sigma[1]), lwd = 2, col = color[1], bty = "n",
      xlab = "x", ylab = "density", xlim = c(0, 150), n = 300)
curve(dnorm(x, mu[2], sigma[2]), lwd = 2, col = color[2], add = T)
curve(dnorm(x, mu[3], sigma[3]), lwd = 2, col = color[3], add = T)
```



Simulating data

We simulate data by drawing dwell times from the dwell-time distribution of the current state and then draw the next state using the conditional transition probabilities. The state-dependent process is drawn conditional on the current state.

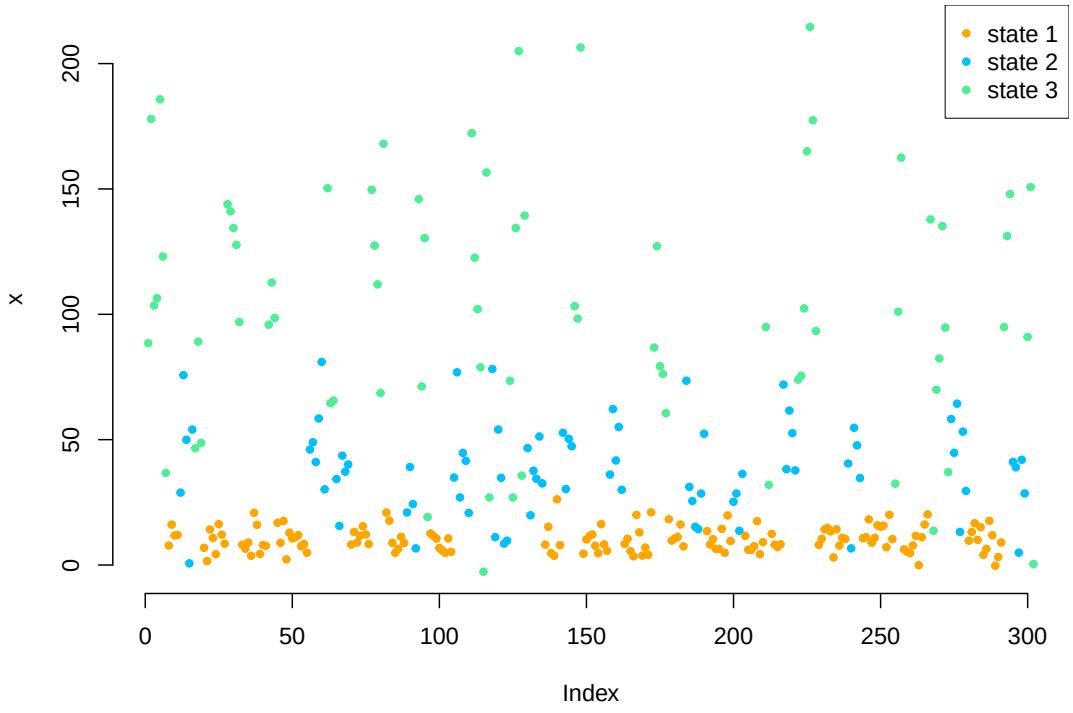
```
set.seed(123)

k = 50 # number of stays
s = rep(NA, k)
s[1] = sample(1:3, 1) # uniform initial distribution
staylength = rpois(1, lambda[s[1]]) + 1 # drawing dwell time from shifted Poisson
C = rep(s[1], staylength)
x = rnorm(staylength, mu[s[1]], sigma[s[1]])

for(t in 2:k){
  # conditionally drawing state
  s[t] = sample(c(1:3)[-s[t-1]], 1, prob = omega[s[t-1], -s[t-1]])
  staylength = rpois(1, lambda[s[t]]) + 1 # drawing dwell time from shifted Poisson

  C = c(C, rep(s[t], staylength))
  x = c(x, rnorm(staylength, mu[s[t]], sigma[s[t]]))
}

plot(x, pch = 20, col = color[C], bty = "n")
legend("topright", col = color, pch = 20,
       legend = paste("state", 1:3), box.lwd = 0)
```



Writing the negative log-likelihood function

We now write the negative log-likelihood function for an approximating HMM. We need the extra argument `agsizes` to specify the aggregate sizes that should be used to approximate the dwell time distributions. These should be chosen such that most of the support of the state-specific dwell-time distributions is covered.

```
mllk = function(theta.star, x, N, agsizes){
  mu = theta.star[1:N]
  sigma = exp(theta.star[N+1:N])
  lambda = exp(theta.star[2*N+1:N])
  if(N>2){
    # this is a bit complicated as we need the diagonal elements to be zero
    omega = matrix(0,N,N)
    omega[!diag(N)] = as.vector(t(matrix(c(rep(1,N),
                        exp(theta.star[3*N+1:(N*(N-2))])),N,N-1)))
    omega = t(omega)/apply(omega,2,sum)
  } else{ omega = matrix(c(0,1,1,0),2,2) }
  dm = list() # list of dwell-time distributions
  for(j in 1:N){ dm[[j]] = dpois(1:agsizes[j]-1, lambda[j]) } # shifted Poisson
  Gamma = Lcpp::tpm_hsmm(omega, dm)
  delta = Lcpp::stationary(Gamma)
  allprobs = matrix(1, length(x), N)
  ind = which(!is.na(x))
  for(j in 1:N){
    allprobs[ind,j] = dnorm(x, mu[j], sigma[j])
  }
  -Lcpp::forward_s(delta, Gamma, allprobs, agsizes)
}
```

Fitting an HSMM (as an approxiating HMM) to the data

```
# intial values
theta.star = c(10, 40, 100, log(c(5, 20, 50)), # state-dependent
               log(c(7,4,4)), # dwell time means
               rep(0, 6)) # omega

agsizes = qpois(0.95, lambda)+1

t1 = Sys.time()
mod = stats::nlm(mllk, theta.star, x = x, N = 3, agsizes = agsizes, stepmax = 2)
Sys.time()-t1
#> Time difference of 0.4898059 secs
```

HSMMs are rather slow (even using C++) as we translate the additional model complexity into a higher computational overhead (31 states here).

Results

```
N = 3
(mu = mod$estimate[1:N])
#> [1]  10.16569  39.06161 107.66034
(sigma = exp(mod$estimate[N+1:N]))
#> [1]  4.78882 19.35639 48.56115
(lambda = exp(mod$estimate[2*N+1:N]))
#> [1] 6.942983 4.595469 3.354765
omega = matrix(0,N,N)
omega[!diag(N)] = as.vector(t(matrix(c(rep(1,N),
                          exp(mod$estimate[3*N+1:(N*(N-2))])),N,N-1)))
omega = t(omega)/apply(omega,2,sum)
omega
#>           [,1]      [,2]      [,3]
#> [1,] 0.0000000 0.5541031 0.4458969
#> [2,] 0.5040938 0.0000000 0.4959062
#> [3,] 0.6654703 0.3345297 0.0000000
```

References

Langrock, Roland, and Walter Zucchini. 2011. "Hidden Markov Models with Arbitrary State Dwell-Time Distributions." *Computational Statistics & Data Analysis* 55 (1): 715–24.

Zucchini, Walter, Iain L. MacDonald, and Roland Langrock. 2016. *Hidden Markov Models for Time Series: An Introduction Using R*. Boca Raton: Chapman & Hall/CRC.