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

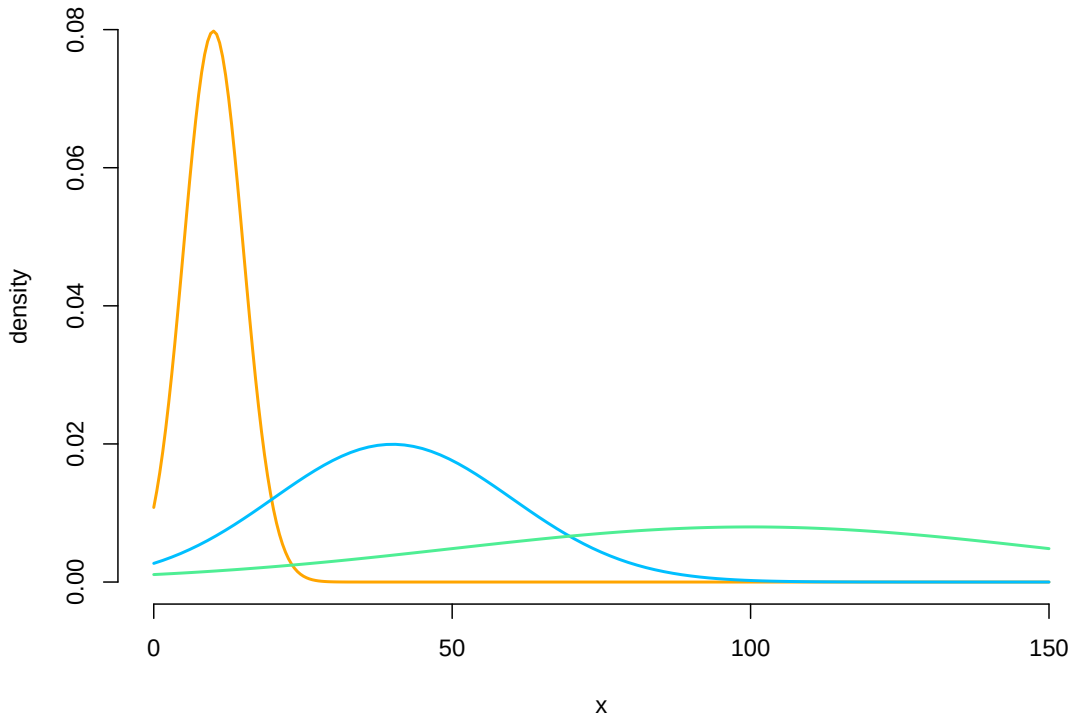
We begin by considering homogeneous HSMMs. In such models, each state has an associated state dwell-time distribution. The transition probability matrix of a regular HMM is replaced by these distributions and the conditional transition probabilities given the state is left.

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.

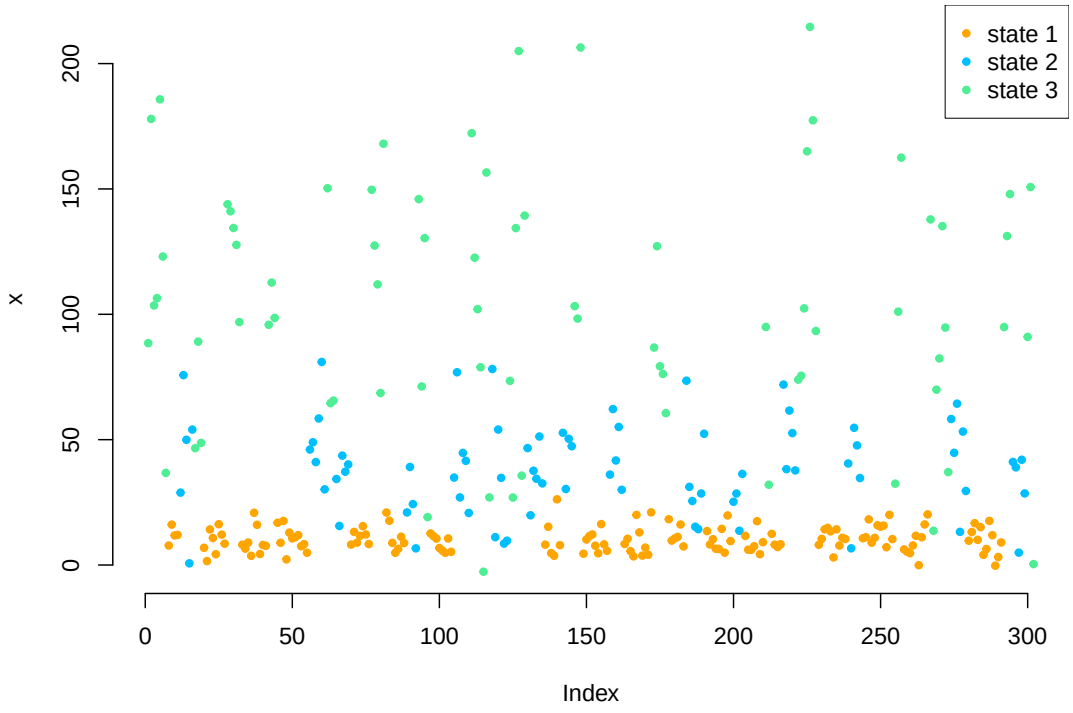
```
set.seed(123)

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. As a semi-Markov chain is specified in terms of state-specific dwell-time distributions and conditional transition probabilities given that the current state is left, we have to compute both (here called `dm` and `omege`). The transition probability matrix of the approxmiating HMM can then be computed by the function `tpm_hsmm()` where the exact procedure is detailed by Langrock and Zucchini (2011). 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[ind], 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 0.4972029 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
```

Real-data application

We now want to briefly show the analysis of a real data set using hidden semi-Markov models. For this purpose we use the movement track of an Arctic muskox contained in the `R` package `PHSMM`. Originally these data where collected by ... and have already been analyzed by ...

```
# install.packages("PHSMM")
data = PHSMM::muskox[1:1000,] # only using first 1000 observations for speed
head(data)
```

```
#>             date tday        x       y       step
#> 88273 2013-10-12   15 513299.2 8264867  17.998874
#> 88274 2013-10-12   16 513283.4 8264875   8.214733
#> 88275 2013-10-12   17 513284.3 8264883   7.205098
#> 88276 2013-10-12   18 513280.4 8264877  53.378332
#> 88277 2013-10-12   19 513252.0 8264922 719.242687
#> 88278 2013-10-12   20 513386.7 8265629  10.797127
```

As these data have already been preprossed, we can immediately write the negative log-likelihood function. When modeling the dwell-time distribution of real processes, it is typically advisable to use a more flexible distribution than the shifted Poisson distribution, as the latter cannot account for overdispersion. Here, we will employ the shifted negative binomial distribution that yields the Poisson distribution as a special case for the dispersion parameter equal to zero. The state-dependent step lengths are modeled by gamma distributions, where we reparametrize the gamma distribution in terms of its mean and standard deviation as opposed to shape and scale for better interpretability.

```
mllk_muskox = function(theta.star, step, N, agsizes){
  # parameter transformation from working to natural
  mu = exp(theta.star[1:N]) # step mean
  sigma = exp(theta.star[N+1:N]) # step standard deviation
  mu_dwell = exp(theta.star[2*N+1:N]) # dwell time mean
  phi = exp(theta.star[3*N+1:N]) # dwell time dispersion
  if(N>2){
    # conditional transition probability matrix
    omega = matrix(0,N,N)
    omega[!diag(N)] = as.vector(t(matrix(c(rep(1,N),
                          exp(theta.star[4*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){
    # R allows to parametrize by mean and size where size = 1/dispersion
    dm[[j]] = dnbinom(1:agsizes[j]-1, mu=mu_dwell[j], size=1/phi[j])
  }
  Gamma = Lcpp::tpm_hsmm(omega, dm)
  delta = Lcpp::stationary(Gamma)
  allprobs = matrix(1, length(step), N)
  ind = which(!is.na(step))
  for(j in 1:N){
    # we reparametrise the gamma distribution in terms of mean and sd
    allprobs[ind,j] = dgamma(step[ind], shape = mu[j]^2 / sigma[j]^2,
                             scale = sigma[j]^2 / mu[j])
  }
  -Lcpp::forward_s(delta, Gamma, allprobs, agsizes)
}
```

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

```
# intial values
theta.star = c(log(c(4, 50, 300, 4, 50, 300)), # state-dependent mean and sd
               log(c(3,3,5)), # dwell time means
```

```
               log(c(0.01, 0.01, 0.01)), # dwell time disperion
               rep(0, 6)) # omega

agsizes = c(11,11,14)

t1 = Sys.time()
mod_muskox = stats::nlm(mllk_muskox, theta.star, step=data$step, N=3, agsizes=agsizes,
                        iterlim = 500)
Sys.time()-t1
#> Time difference of 3.546046 secs
```

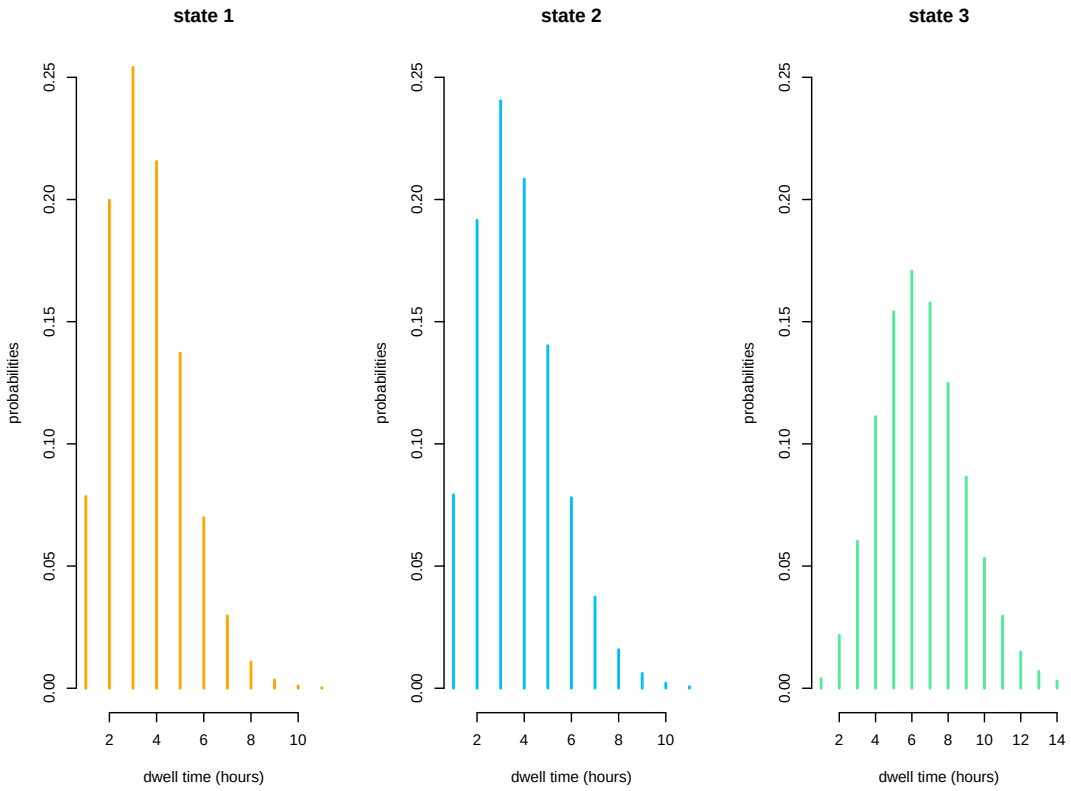
Results

We retransform the parameters for interpretation

```
theta.star = mod_muskox$estimate; N = 3
(mu = exp(theta.star[1:N])) # step mean
#> [1]   4.408109  55.515891 306.504787
(sigma = exp(theta.star[N+1:N])) # step standard deviation
#> [1]   3.148127  50.337602 331.539171
(mu_dwell = exp(theta.star[2*N+1:N])) # dwell time mean
#> [1] 2.544975 2.660335 5.541752
(phi = exp(theta.star[3*N+1:N])) # dwell time dispersion
#> [1] 3.643785e-05 3.751638e-02 1.836526e-09
omega = matrix(0,N,N)
omega[!diag(N)] = as.vector(t(matrix(c(rep(1,N),
                        exp(theta.star[4*N+1:(N*(N-2))])),N,N-1)))
omega = t(omega)/apply(omega,2,sum)
omega
#>           [,1]      [,2]         [,3]
#> [1,] 0.0000000 0.6865910 3.134090e-01
#> [2,] 1.0000000 0.0000000 2.959219e-08
#> [3,] 0.8210834 0.1789166 0.000000e+00
```

In this case the Poisson distribution would have been suffficiently flexible, as all dispersion parameters were estimated very close to zero. We can easily visualize the estimated state-specific dwell-time distributions:

```
par(mfrow = c(1,3))
for(j in 1:N){
  plot(1:agsizes[j], dnbinom(1:agsizes[j]-1, mu=mu_dwell[j], size = 1/phi[j]),
       type = "h", lwd = 2, col = color[j], xlab = "dwell time (hours)",
       ylab = "probabilities", main = paste("state",j), bty = "n", ylim = c(0,0.25))
}
```



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.