HMM Summary

Camrin Braun and Ben Galuardi

October 11, 2016

The Approach

We present a process-based approach to estimate residency and behavior from uncertain and temporally correlated movement data collected with electronic tags. To do this, we use a state-space model (SSM) for which the animal tracking problem is governed by two parts. The system process describes the animal movement and behavior, and the observation model links the process (i.e. movements) to the data. Inference regarding the unobservable system process can then be established via this link using statistical methodology (filtering) which updates location and behavior estimates with observed data. The system process estimation problem is formulated as a hidden Markov model (HMM) on a spatial grid in continuous time. Using the grid to explicitly resolve space, location estimation can be supplemented by or based entirely on environmental data (e.g. temperature).

Model formulation

The movement of an animal in continuous time is a (biased) brownian motion in the longitudinal (x) and latitudinal (y) direction. Given the current behavior state I_t of the animal, the Brownian motion is decribed by a drift $u_{I_t} = (u_x, u_y)_{I_t}$ with unit km day^{-1} and a diffusivity matrix D_{I_t} with unit km^2 day^{-1}.

To proceed with the analysis of the joint process of movement and behavioral shifts, we introduce the probability density $\phi_{\mathrm{i}}(x,y,t)$ which describes the probability that the animal at time t is located at (x, y) and is in behavioral state i. This can be formulated as a partial differential equation (PDE) describing the time evolution of ϕ_{i} as an advection-diffusion equation that includes behavior switching:

$$\frac{d\phi_i}{dt} = -\nabla \times (u_i \phi_i - D_i \nabla_{\phi_i}) + \sum_j \lambda_{ji}\ \phi_j$$

where ∇ is the two-dimensional spatial gradient operator. The advection and diffusion terms (in parentheses) describe the flow of probability between regions in space. The behavior switching (summation term) is a weighted sum over the switching rates that represents probability of flow between behavioral states.

To solve Eq. 1 some form of numerical approximation is required. Our approach discretizes the continuous state–space into a finite, albeit large, number of uniformly sized squares (states) (Thygesen et al. 2009). Using this discretization of the state-space, ϕ is no longer a probability density, but is instead represented by a vector j containing the state probabilities ϕ_α, where the state index $\alpha = (x, y, i)$ is composed of location in x and y and the behavior state i.

Filtering

A HMM filter provides the probability distribution of the states forward in time conditional on data, $\phi(t_k|Z_k)$. These state estimates are calculated successively by alternating between so-called time and data updates of the current state.

Smoothing

Finally, the recursions of the HMM smoothing step work backwards in time using the filtered state estimates and all available data to determine the smoothed state estimates, $\phi(t_k|Z_N)$. The smoothed state estimates are more accurate and generally appear 'smoother' than the filtering estimates because they exploit the full data set (Z_N). When fitting an SSM in a Bayesian context the smoothing step provides the posterior distribution of the state. The probability distribution of all states at specific times are the state estimates returned from the HMM smoothing algorithm.

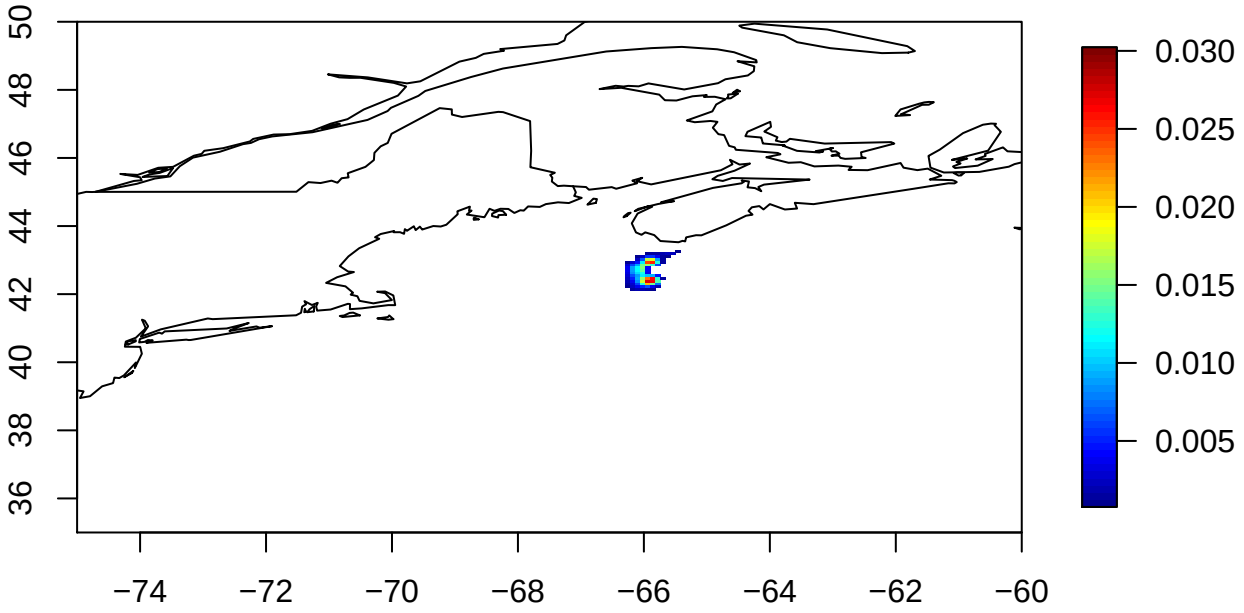

Figure 1: An example of the posterior distribution of state 2 at time 2.

Parameter Estimation

Maximum likelihood

In addition to the state estimates, the filter mentioned above returns a likelihood measure which indicates how well the model fits the data. Thus, the likelihood function, L, of the unknown parameters θ (e.g. diffusion, switching rates) can be evaluated at, say, θ_0 by running the filter using the parameter values in θ_0. Maximum likelihood (ML) estimation of model parameters is then straight forward and can be performed by most standard numerical optimizers. However, recent work by Woillez et al (2016) further exploited the properties of the discrete setting and addressed a joint ML estimation of all model parameters using an iterative Expectation-Maximization (EM) framework.

Expectation maximization

The great interest of the EM algorithm is that it delivers a two-step iterative algorithm: the E-step computes the posterior distribution $P((X_t)_{t=0:N-1}|(Y_t)_{t=0:N-1})$ given current model parameter estimates; and the M-step updates model parameters according to a ML criterion reweighted by the posterior distribution. Woillez et al (2016) took the EM one step further because of sensitivity to initial values. Thus, they considered a stochastic version of the EM (SEM) where it replaces the numerical evaluation of the posterior by a sampling step. Overall, the SEM procedure involves two steps: [1] the E-step comes to sampling N_{SEM} trajectories $(X^{(i)})_{i=1:N_{SEM}}$ from posterior $P(X|Y,\Theta^{(v)})$ using the standard forward-backward HMM procedure, where

$\Theta^{(v)}$ refers to model parameter(s) and could include diffusion, advection, transition probability, etc. [2] the M-step comes to updating estimate $\Theta^{(v+1)}$. This two-step procedure is iterated until some convergence threshold is met.

Implementing the EM

We'll start with the classic EM approach (non-stochastic) in which the E-step is simply the computation of the posterior distribution given current model parameter estimates (Fig. 1). The M-step "updates model parameters according to a ML criterion reweighted by the posterior distribution."(from Woillez)

In our case, $\Theta^{(v)}$ represents diffusion (for 2 states) and transition probability. The process looks like:

```
# START WITH INITIAL PARAMETER VALUES
D1 = c(8.908, 10.270) # mean, SD for migratory state
D2 = c(3, 1) # mean, SD for resident state
p = c(.707, .866) # switch probability

# GENERATE DIFFUSIVE KERNELS
K1 = as.cimg(gausskern(D1[1], D1[2], muadv = 0))
K2 = as.cimg(gausskern(D2[1], D2[2], muadv = 0))
P <- matrix(c(p[1], 1-p[1], 1-p[2], p[2]), 2, 2, byrow = TRUE)

# RUN THE FILTER STEP
f = hmm.filter(g, L, K1, K2, P)

# RUN THE SMOOTHING STEP
s = hmm.smoother(f, K1, K2, P, plot = F)

#------------------------#

# UPDATE D
# need to calculate new D from posterior distribution, s
# see Questions section below

#------------------------#

# UPDATE P
# get states from s
sv <- apply(s[1,,,], 1, sum) > apply(s[2,,,], 1, sum)
sv <- -sv + 2

# difference it to find transitions
svd = rbind(sv, c(0, diff(sv)))

# get a new transition probability matrix
r1 = rev(table(svd[2, svd[1,] == 1]) / sum(svd[1,] == 1))
r2 = rev(table(svd[2, svd[1,] == 2]) / sum(svd[1,] == 2))
P = matrix(rbind(r1, r2), 2, 2)

#------------------------#

# CALCULATE CHANGE IN PARAMETER VALUES OVER LAST SEVERAL ITERATIONS
# Woillez used "ratio betwen the average over the last 20 values of D and the new value
# of D below 1%"
```

```
#------------------------#

# IF THRESHOLD IS NOT MET, RE-ITERATE
# now have a new set of diffusion parameters (D1, D2) and switching probability (P) and
# we can iterate until we cross some threshold of change in parameters

#------------------------#
```

The Question

The process portion of the HMM is represented by diffusive kernels K_1 and K_2 based on diffusion parameters in D (see script above). These parameters are of differing bandwidth and extent estimates for 'migratory' and 'resident' states (Fig. 2).

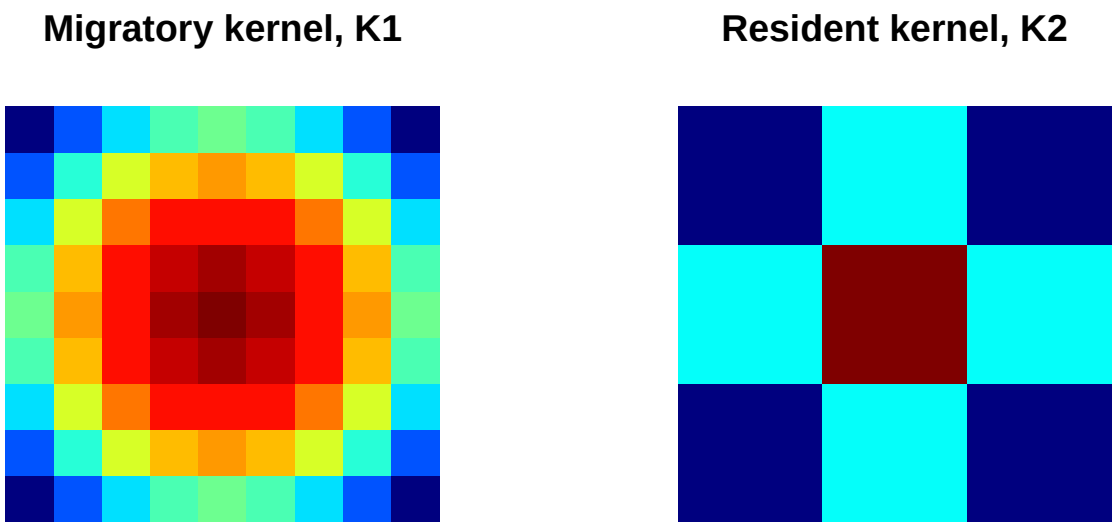

Figure 2: Movement kernels using fixed diffusion rates of 8.908 (+/- 10.27) and 3 (+/- 1) for [A] migratory and [B] resident movement states, respectively.

To run an Expectation-Maximization process, we need to be able to calculate new parameters based on our posterior output in an iterative fashion. Our inputs are K_1, K_2 and P_1, representing our resident, migratory kernels (Fig. 2) and transition probability matrix:

$$\begin{pmatrix} 0.707 & 0.293 \\ 0.134 & 0.866 \end{pmatrix}$$

Calculating a new P_n is straightforward (see script above). What we need help with is determining kernel parameters based on our output. In other words, given a posterior probability grid (Fig. 1), how do we calculate kernel bandwidth and extent to derive new K_1 and K_2? All literature so far has pointed to the standard process of determing a kernel based on bandwidth, etc., not the other way around.