Statistical structure of locomotion and its modulation by odors

Liangyu Tao, Siddhi Ozarkar, Jeffrey M Beck, Vikas Bhandawat*

*Correspondence should be addressed to V.B (vb468@drexel.edu)

User manual for running mixture models of HHMM and HMMs

The software was written using MATLAB 2017.

Data location:
The archived dataset of fly track excel sheets are in dryad: here
The processed dataset and model used in the paper are on Dropbox: here

Input data format:
In the current setup of the "mixHHMMcaller.m" code, the data should be in a 1 x nFly cell array called "data". Each cell contains the data of one continuous trial (referred to as fly moving forward since flies were used in the paper). Each cell in the matrix should be a nxm double matrix, where each row represents an observable (i.e. vpar and vperp) over time (column).

* note that nFly refers to the number of trials/animals/flies in the dataset

HHMM parameters:

1. NC: number of potential models to fit to the dataset
2. Qdim: number of high-level (HL) states
3. Dim: number of low-level (LL) states per HL state
4. D: dimensionality of the input observables (2 = 2 observables (vpar and vperp))
5. obsTypes: a cell matrix with the following properties
    i. obsTypes{i}.idx = the row index corresponding to the observables assigned to this obsTypes.
    ii. obsTypes{i}.dist = the distribution that you want to represent the observables as for the lower level states.

Distribution	String name	Number of dimensions
Multivariate Normal	'mvn'	2+
Normal	'normal'	1
Gamma	'gamma'	1

* note that we can use different distribution types for different observables (i.e. normal for speed and gamma for curvature). To implement this, you would set the first cell of obsTypes to the idx of speed and the dist to 'normal' and the second cell of obsTypes to the idx of curvature and dist to 'gamma'. This implementation has not been teted. In the paper, we used bivariate normals for our observables.

Initializing a model:
To initialize a mixture model of HMM, run the following line:

model =
mixHMM(NC,dim*Qdim,D,obsTypes,ones(NC,1),eye(Qdim*dim)+ones(Qdim*dim,Qdim*dim)/Qdim/dim
,ones(Qdim*dim,1)/Qdim/dim);

To initialize a mixture model of HHMM, run the following line:
model = mixHHMM(NC,Qdim,dim,D,obsTypes,ones(NC,1),eye(Qdim)+ones(Qdim)/Qdim,
ones(Qdim,1)/Qdim,eye(dim)+ones(dim)/dim,ones(dim,1)/dim);

To initialize a single HMM model, you can do one of the following

1. Initialize the same way as the mixHMM, but with NC = 1
2. Initialize using the HHMM:
   model = HMM(dim,D,obsTypes,eye(Qdim)+ones(Qdim)/Qdim,
   ones(Qdim,1)/Qdim,eye(dim)+ones(dim)/dim,ones(dim,1)/dim);

To initialize a single HHMM, you can call do of the following

1. Initialize the same way as the mixHHMM, but with NC = 1
2. Initialize using the HHMM:
   model = HHMM(Qdim,dim,D,obsTypes,eye(Qdim)+ones(Qdim)/Qdim,
   ones(Qdim,1)/Qdim,eye(dim)+ones(dim)/dim,ones(dim,1)/dim);

* Note that here, we are initializing with a prior that favors self-transitions. You can initialize with whatever prior you wish

Commonly used methods during fitting:

mixHHMM and mixHMM methods

1. Model.initialize(data, iters)
2. Model.Update(data,iters,modeliters)
3. Model.prune()
4. Model.split(data,modeliters,idx1)
5. Model.fillunused(data,modeliters,NC2f)
6. Model.plotclusters(data,fignum)

HHMM and HMM methods

1. Model.initialize(data)
2. Model.update(data)

Where:

data = cell array of data used to fit/update the model parameters
iters = number of iterations for updating fly assignments for mixHHMM and mixHMM models
modeliters = number of iterations for updating model parameters based on fly assignments
idx1 = model cluster to split into new models
NC2f = number of empty clusters to fill
fignum = figure number to plot to

Model output:

The result will be an object called “model” of class HHMM. Note that to access this model and its properties, you must have the folder with the class files in your path.

mixHHMM properties

1. Qdim, dim, D, and obsTypes are the same as the input parameters
2. NC = number of final models
3. HHMMs = 1xNC cell array of each of the final HHMM models
    i. Qdim, dim, D, and obsTypes are the same as the input parameters
    ii. Q = HL state transition probability
        a. Q.alpha_0 = prior/initial guess before model fit
        b. Q.alpha = final model fit
    iii. Qpi0 = initial HL state distribution
        a. Qpi0.alpha_0 = prior/initial guess before model fit
        b. Qpi0.alpha = final model fit
    iv. A = 1x Qdim cell array of LL state transition probability
        a. Each cell has the same format as Q, but for LL states

        pi0 = 1 x Qdim cell array of initial LL state distributions for each HL state
        b. Each cell has the same format as Qpi0, but for LL states
    v. Aidx = 1 x Qdim cell array of LL state index belonging to each HL state
    vi. obsModels = distribution fits for each LL state in the observable space (dim x Qdim cell array)
        a. types = same as the obsModels
        b. dists{i} = parameters of distribution fits (i.e. covariance, mean, etc)
    vii. HMMequiv = the equivalent HMM model for the HHMM model
    viii. p = 1 x nFly cell array of probability of being in a given LL state (rows) over time (columns)
    ix. Qp = Same as p, but for probability of being in a given HL state
4. pi = distribution of fits for each model
    i. pi.dim = number of models
    ii. pi.alpha_0 = initial model assignments (sums to 1)
    iii. pi.alpha = final model assignments (sums to 1+number of flies)
5. p = NC x nFly probability fit of each fly to each model. Each column sum to 1, where 1 means the fly is best fit to the given model
6. NA = Number of flies that is best fit to each model (sum of each row of p)
7. L = lower bound

The mixHMM, HHMM, and HMM properties will be in a similar format.

<u>Other notes:</u>

1. Depending on your dataset, we found that sometimes, it might be useful to preprocess the observables that you use using z-scoring. But you can run it on the raw observables first to get an idea of the model fit.
2. Fitting the mixHHMM and mixHMM is computationally intensive. It will take a few hours to a day to run with reasonable number of iterations. Large RAM and SSD drive are near essential.
    i. This is because each model is fit sequentially (not in parallel) and that the code computes a model fit of all flies to each model.
3. Fitting a single HHMM or HMM to all the flies is much faster

4. One workaround if you want to fit a mixHHMM or mixHMM to a dataset with many flies/animals/trials etc is to first cluster each fly's tracks using something like kmeans. Then, initiate each HHMM of the mixHHMM by fitting to the corresponding clusters of flies (1 HHMM/cluster)