HMM-VAR Input/Output structures

Preprocessing

Dimensional reduction for MAR

The space in which the MAR is trained is the original space when the number of dimensions is reasonable (e.g. hippocampus data, with ~10 recording points). For MEG data (~4.000) we retain the e.g. 40 first principal components; spectral contents are estimated after projecting the MAR parameters back to the original dimensions.

Note: the MxN projection matrix of the PCA is often called `loadings` in the code. It should be replaced with the NxN identity if no reduction is necessary.

The state matrix

Probabilistic assignments (responsibilities, Gamma)

The matrix of probability of state k at time `T;` in the simplest case of just on trial, we have size(Gamma) = `(T-order) x K`.

Typically, MEG runs have very skewed assignments with one state absorbing most of the probability. For electrophysiological data this is not the case.

Hard assignments (trajectory, ViterbiGamma)

Hard assignments can be obtained using the Viterbi algorithm (Rabiner 1989), implemented in `hmmdecode.m`. They are appropriate for interpretability and for summarized representations (e.g. coloring the trace according to state).

Residuals

We train on residuals from a global MAR of desired order orderGL in order to focus on the transient patterns of the data. This is different from pre-whitening, because here the model attempts to predict the residuals of the global MAR as a linear function of the **original** time series. It is in this sense that our MAR is not strictly an *auto*regression.

So, if we assume the data to be generated as

where are the state-dependent (local) MAR coefficients and are the global MAR coefficients. Here would be the residuals of the global MAR fit.

At the end, spectral quantities must be extracted from a last MAR pass on the original data, so as to reflect its full (not residual) structure.

Unsupervised and (semi)supervised learning

The software can be used both for unsupervised and (semi)supervised learning. In the former case, we let the HMM engine to freely segment all the data into K states. In the latter case, we fix the state assignment using available class information for some of the data and we let the HMM engine infer the states or classes for the unlabelled part of the data.

Embedded estimation

The “embedded” approach expands the data such that each time point includes the value of the signal at this time point and also the value of the signal from neighboring points

Spectral estimation

There are two ways to perform spectral estimation: the parametric and the non-parametric way. The non-parametric estimation is recommended. The parametric estimation uses the coefficients of the MAR model, either as they come from the HMM-MAR inference algorithm, or computed by maximum likelihood (via the function `mlhmmmar.m`). Since regularisation introduces a bias, it might be desirable to obtain the spectral information from the maximum-likelihood estimation. The specification of the function is

```
function [fit] = hmmspectramar(hmm,options)
```

where options is structure with fields

`loadings` - in case of a PCA transformation, this is the PCA loading matrix

`Fs` - sampling rate of data, in Hz

`Nf` - number of frequency points to be computed (resolution)

`fpass` - interval of frequencies in which the spectral information is retrieved

This function stores for each state, a number of spectral quantities in a the list fit. Each fit.state(k) contains, in addition of the frequency vector `f` (`Nf x 1`) the following Nf x N x N quantities (Faes and Nollo 2011):

psd	Power Spectral Density matrix
ipsd	Inverse Power Spectral Density matrix
coh	Coherence matrix
pcoh	Partial Coherence matrix
pdc	Baccala's Partial Directed Coherence
phase	Phase matrix

The non-parametric estimation uses the multitaper, weighting the data with the posterior state probabilities. The Partial Directed Coherence is an approximation that uses the Wilson's algorithm for spectral matrix factorization. The specification of the function is

```
function [fit] = hmmspectramt(X,T,options)
```

where options is structure with fields

Fs - sampling rate of data, in Hz

win - duration (number of time points) of non-overlapping window. This controls the frequency resolution

tapers - A numeric vector [TW K] where TW is the time-bandwidth product and K is the number of tapers to be used (less than or equal to 2TW-1).

fpass - interval of frequencies in which the spectral information is retrieved

p - p-value for computing jackknife confidence intervals (default 0)

numIterations - no. iterations for the Wilson algorithm (default: 100)

tol - tolerance limit (default: 1e-18)

Gamma - posterior probabilities of the states (default, a column of 1)

rlowess - if 1, smoothing using the robust lowess procedure is applied to the spectral decomposition, coherence and PDC

removezeros - if 1, those time points that has Gamma(t,k)=0 are removed

pad - padding factor for the FFT (defaults to 0)

In addition to the information returned by hmmspectramar, it returns, for each state:

psderr	Error for the Power Spectral Density matrix
coherr	Error for the coherence
pdcerr	Error of the PDC
sdphase	Jackknife standard deviation of the phase

The errors are computed only if options.p>0

Basic training

This is done by the function *hmmmar.*

Inputs

The function *hmmmar* receives three inputs: *data*, *T* and *options*.

data can be either a matrix of signals with one row per time point and one column per channel, or a struct with two fields: *X*, with the matrix of signals, and *C*, a matrix with as many rows as time points and as many columns as number of states, containing the class probability assignments

that we want to pre-specify. The HMM engine will infer the states/classes corresponding to those elements of C that are set to NaN. It is assumed that whole rows will set to NaN. In case the data is presented in trials, subjects or epochs, data will just contain the concatenation of such epochs.

T is a vector containing the number of data points for each trial/subject/epoch. If there is only one batch, then T contains just a scalar with the total number of data points.

The structure *options*, which circulate in the code under the name `hmm.train`, may contain the following fields

`K` - maximum allowable number of HMM states (mandatory, with no default)

`Fs` - Sampling frequency (default to 1)

`order` - order of the MAR model regressing the residuals on the original data. If zero, an HMM with Gaussian observations is trained (mandatory, with no default).

`covtype` - whether the covariance matrix of the noise is the same for all states (unique) and whether it is is diagonal (diag). Diagonal matrices **cannot** be used to run on principal components.

- **`full`** - a full covariance matrix for each state
- **`uniquefull`** - one full covariance matrix for all states
- **`diag`** - a diagonal full covariance matrix for each state
- **`uniquediag`** - one diagonal covariance matrix for all states

`orderGL` - order of the global MAR for computing the residuals. If zero, run the HMMMAR directly on the original data. Default is zero

`lambdaGL` - L^2 regularization parameter to be applied to the global MAR for computing the residuals. If zero, the global MAR is computed by maximum-likelihood. Default is zero.

`timelag` - sets how many points to skip when regressing on the past, e.g. timelag=2 skips one sample for each sample that is taken, time_lag=3 skips 2, etc… if timelag>1, order represents how far in time the MAR model looks at. For example: with order 5 and timelag=2, the MAR regression will use samples -1, -3 and -5 (provided the present sample is represented by 0). Default = 1.

`embeddedlags` - for the embedded approach, we will specify *`embeddedlags>0`* to embed up to these number of time points at each single time point (see above). The default is 0, which corresponds to no embedding.

`tol` - minimum relative decrement of free energy to consider the algorithm converged and stop. Default is 0.001

`initcyc` - number of iterations for the states probabilities initialisation algorithm. If 0, then the states probabilities are initialised at random. Default is 5.

`cyc` - *maximum* number of variational EM cycles to run the HMM-MAR. The algorithm with stop earlier if `tol` is reached. Default is 1000

repetitions - The variational inference algorithm is not guaranteed to find the globally best solution. To get around this, we can run the algorithm more than once by specifying repetitions>1, keeping the one with the best free energy. Default is 1.

whitening - whether or not a prewhitening process is done on the data, so that the global covariance matrix is the identity. Default is 0

keepS_W - whether to keep or not the covariance matrices of the autoregression coefficients. Use only to save disk space when the number of channels is too high. Defaut is 0.

Gamma - initial estimate for the states probabilities (optional)

verbose - whether to show algorithm progress

Outputs

hmm - structure that contains HMMMAR output.

K - number of ***remaining*** states after training - could have been shrunk from the initial, which is stored as hmm.train.K. Training can shrink the number of states so that hmm.Khmm.train.K.

P - expected KxK transition matrix between states, with elements in [0,1].
Pi - expected Kx1 initial state vector probabilities, with elements in [0,1].

Dir2d_alpha - KxK matrix of Dirichlet concentration parameters that models P
Dir_alpha - Kx1 vector of Dirichlet concentration parameters that models Pi

prior - hyperpriors for each parameter in the model
- Dir2d_alpha - KxK matrix Dirichlet...
- Dir_alpha - Kx1 vector Dirichlet...

state - parameters in the observation model
- Omega - covariance matrix of the noise
- W - state-specific VAR output Kx1 vector of HDF references
  - Mu_W - Np X N matrix of expected MAR coefficients, where N is the number of channels
  - S_W - Variance of the MAR coefficients (either N^2 p x N^2 p, or (N x Np x Np, depending on the type of covariance matrix - full or diagonal)
- alpha -
- constraint -
- prior -
- sigma -

train - input algorithm options

Gamma - the probabilistic estimation of states, a `K x (T-order)` matrix of [0,1] elements.

Xi - joint probability of past and future states conditioned on data

vpath - The Viterbi path (“hard” assignments), a `(T-order) x 1` vector of elements in `1..K`.

GammaInit - The initial probabilistic estimation of states

residuals - if the model is trained on the residuals, the value of those

fehist - The historic of the free energy across iterations - the last value corresponds to the actual free energy of the model

Example inputs (I’d remove this)

Typical values for MEG full brain vs. electrophysiology data

		MEG data	ephys tetrode data
preprocessing			
PCA dim. reduction	PCAxx	40	not necessary
options.			
number of states	K	8	10
sample rate	Hz	100	667
stop criterion	tol	1E-5	1E-5
hmm cycles	cyc	300	500
embedded lags	embeddedlags		
global MAR order	orderGL	2	0, **10 .. 48**
main MAR order	order	4	10
ML MAR order	orderML	6	10
global regularisation	lambdaGL		
covs of states	covtype	‘full’	‘full’
Init cycles	initcyc	5	5
Initial states	Gamma	[]	[]
data.			
length/batches	T		5.2E5 (13’ session at 667Hz)
channels	ndim	40	1-10
code constants			

winsorization percentile	`winsor`		99 (data-dependent)
prior cov. rate	`priorcov_rate`		
prior cov. shape	`priorcov_shape`		
observation model	`obsmodel`		

Example HMM-MAR function call

```
options = struct('K',8);
options.cyc = 100;
options.tol = 0.00001;
options.initcyc = 10;
options.order =  20;
options.covtype = 'diag';
[hmm, Gamma, Xi, vpath, GammaInit, residuals, fehist] = hmmmar (data,T,options)
```

Cross-validation

There are two main methods for model selection. One is to select the combination of parameters that with the minimal free energy. The other is to use cross-validation. For each left-out fold, we would fix the states estimated on training, estimate the posterior probabilities of the states over the left-out fold, and get the squared prediction error for this fold. The average error across channels is computed, weighting each channel error by the channel variance. This procedure is performed by the function `cvhmmmar`. This function has similar inputs parameters than `hmmmar`, but there are a few differences.

Inputs

The function cv*hmmmar* receives three inputs: *data*, *T* and *options*. The same considerations for data and T that we have for `hmmmar` hold for `cvhmmmar`. The structure *options* may contain the following fields

`K` - maximum allowable number of HMM states (mandatory, with no default)

`Fs` - Sampling frequency (default to 1)

`order` - order of the MAR model regressing the residuals on the original data. If zero, an HMM with Gaussian observations is trained (mandatory, with no default).

`covtype` - whether the covariance matrix of the noise is the same for all states (unique) and whether it is is diagonal (diag). Diagonal matrices **cannot** be used to run on principal components.

- **`full`** - a full covariance matrix for each state
- **`uniquefull`** - one full covariance matrix for all states
- **`diag`** - a diagonal full covariance matrix for each state
- **`uniquediag`** - one diagonal covariance matrix for all states

orderGL - order of the global MAR for computing the residuals. If zero, run the HMMMAR directly on the original data. Default is zero

lambdaGL - L^2 regularization parameter to be applied to the global MAR for computing the residuals. If zero, the global MAR is computed by maximum-likelihood. Default is zero.

timelag - sets how many points to skip when regressing on the past, e.g. timelag=2 skips one sample for each sample that is taken, time_lag=3 skips 2, etc… if timelag>1, order represents how far in time the MAR model looks at. For example: with order 5 and timelag=2, the MAR regression will use samples -1, -3 and -5 (provided the present sample is represented by 0). Default = 1.

embeddedlags - for the embedded approach, we will specify *embeddedlags*>0 to embed up to these number of time points at each single time point (see above). The default is 0, which corresponds to no embedding.

tol - minimum relative decrement of free energy to consider the algorithm converged and stop. Default is 0.001

initcyc - number of iterations for the states probabilities initialisation algorithm. If 0, then the states probabilities are initialised at random. Default is 5.

cyc - *maximum* number of variational EM cycles to run the HMM-MAR. The algorithm with stop earlier if `tol` is reached. Default is 1000

cvfolds - Either a number indicating the number of folds to compute the cross-validated mean squared prediction error (on trials), or a vector with each index indicating to which fold each trial belongs.

cvrep - The variational inference algorithm is not guaranteed to find the globally best solution. To get around this, we can run the algorithm more than once by specifying cvrep>1, keeping, for each fold, the one with the best free energy. Default is 1.

whitening - whether or not a prewhitening process is done on the data, so that the global covariance matrix is the identity. Default is 0

Gamma - initial estimate for the states probabilities (optional)

verbose - whether to show algorithm progress

Outputs

The function `cvhmmmar returns:`

mcverror - the averaged cross-validated mean squared error

cverror - the cross-validated mean squared error per fold. Each fold is weighted by the amount of testing data points in it.

References

Faes, L, and G Nollo. 2011. “Multivariate Frequency Domain Analysis of Causal Interactions in Physiological Time Series.” *Biomedical Engineering. Trends in Electronics*. cdn.intechopen.com. http://cdn.intechopen.com/pdfs/12918/InTech-Multivariate_frequency_domain_analysis_of_causal_interactions_in_physiological_time_series.pdf.

Fox, E, E B Sudderth, and M I Jordan. 2011. “Bayesian Nonparametric Inference of Switching Dynamic Linear Models.” *Signal Processing, IEEE*. ieeexplore.ieee.org. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5680657.

Rabiner, L. 1989. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.” *Proceedings of the IEEE* 77 (2). ieeexplore.ieee.org: 257–86.