MATLAB Toolbox for Dynamic Stochastic Blockmodels on Network Data (Version 0.2)

Kevin S. Xu and Alfred O. Hero III
University of Michigan
January 23, 2015

Introduction

This toolbox contains a MATLAB implementation of the dynamic stochastic blockmodel (DSBM) proposed in the paper (Xu & Hero III, 2014). It can be used to fit DSBMs to dynamic network data for analysis of network structures and temporal dynamics. For a description of the models and inference algorithms, please see the paper that accompanies the toolbox (Xu & Hero III, 2014).

The toolbox currently includes the following:

- Implementations of the DSBM inference procedures for a priori and a posteriori blockmodels and the local search inference procedure for a posteriori blockmodels.
- Demo of DSBM applied to simulated network data sets.
- Demo of DSBM applied to dynamic email network constructed from Enron corpus (Priebe, Conroy, Marchette, & Park, 2009).

Usage

The inputs and outputs for each function are documented in the function header. For most variables, the last dimension in the array denotes the time step. Adjacency matrices are stored as $n \times n \times T$ arrays, where n denotes the number of nodes, and T denotes the number of time steps. Class memberships are stored in an $n \times T$ matrix.

A priori blockmodels

In the a priori blockmodel setting, class memberships are known or assumed, and the objective is to estimate the time series of edge probability matrices $\Theta^{1:t}$, where θ_{ab}^{t} denotes the probability of forming an edge between nodes in classes a and b at time step t. For a priori blockmodeling, the time series of block density matrices $Y^{1:t}$ is a sufficient statistic for estimating the edge probability matrices $\Theta^{1:t}$. Near-optimal estimates of the edge probability matrices in the maximum a posteriori (MAP) sense can be obtained using the `ekfDsbm` function, which performs extended Kalman filter-

ing on the time series of block density matrices $Y^{1:t}$. $Y^{1:t}$ can be obtained from the time series of adjacency matrices $W^{1:t}$ and class membership vectors $\mathbf{c}^{1:t}$ using the function `calcBlockDens`.

A posteriori blockmodels

In the a posteriori blockmodel setting, class memberships are estimated along with the edge probability matrices. The estimates of the class memberships and edge probability matrices can be obtained using the `ekfDsbmLocalSearch` function, which alternates between applying a local search over the class memberships and applying the extended Kalman filter (EKF) on the block densities for the current class membership estimates. `ekfDsbmLocalSearch` operates directly on the time series of adjacency matrices $W^{1:t}$.

State representation

For undirected graphs, the number of states is $p = k(k+1)/2$, where k denotes the number of classes. For directed graphs, $p = k^2$. Many of the toolbox functions involve manipulation of $k \times k$ matrices, which are symmetric for undirected graphs and asymmetric for directed graphs. We often represent these matrices by vectors of length p, which is the typical means for representing states in dynamic systems. The function `blockmat2vec` converts these quantities from matrix to vector form, while the function `blockvec2mat` does the reverse.

Setting hyperparameters

Four hyperparameters need to be set for both the `ekfDsbm` and `ekfDsbmLocalSearch` functions:

- `initMean`: The mean of the initial state
- `initCov`: The covariance matrix of the initial state
- `obsCov`: The covariance matrix of the observation noise
- `transCov`: The covariance matrix of the process noise driving the state dynamics

The hyperparameters `initMean` and `initCov` constitute the Gaussian prior on the initial state at time 0. The first observation is generated at time 1, so it has covariance given by `initCov` + `transCov`. If no prior knowledge on the initial state is available, set `initMean` to `[]` and `initCov` to a diagonal matrix with large diagonal elements, e.g. 100, to minimize the effect of the prior. It is important to set `initMean` to `[]` and not choose it arbitrarily because this affects the linearization point in the EKF. If you find that your edge probability estimates are unstable, especially towards the beginning of the data, a stronger prior is probably necessary, i.e. decrease the variances in `initCov`.

The hyperparameter `obsCov` denotes the covariance of the observation noise and is a function of the edge probabilities in each block at the current time step. It can be selected automatically by using a plug-in estimate from the EKF predicted state. To use this estimate, set `obsCov` to `[]`.

The final hyperparameter `transCov` relates to the state dynamics. We have not provided a method to automatically select `transCov`. The function `generateStateCov` will generate a covariance matrix for the state transition process noise given two parameters: the variance of the process noise and the covariance of the process noises for any two neighboring blocks. The covariances for all

non-neighboring blocks is taken to be 0. As shown in the paper (Xu & Hero III, 2014), the estimation error of the EKF is not overly sensitive to the choice of `transCov` so it does not have to be chosen as carefully as the other hyperparameters; if additional fine-tuning is desired, one can check the forecasting error of the block densities using the EKF predicted states as a measure of goodness-of-fit.

Optional inputs

Optional inputs (parameters) are specified using the struct `Opt`, which is an input to many functions. The optional inputs are specified as fields of `Opt`. Not all optional inputs are used in all functions; see the function headers for the applicable optional inputs. The list of all optional inputs, with default values, are as follows (in alphabetical order):

- `'classInit'`: Class estimates to initialize local search at initial time step $t = 1$ for a posteriori blockmodeling. If not specified (set to `[]`), spectral clustering will be performed to obtain the initial class estimates. Default: `[]`
- `'directed'`: Whether the networks are directed(set to true or false). *Default:* `true`
- `'eigShift'`: Amount to shift eigenvalues of estimated state covariance matrix if positive definiteness is lost due to numerical rounding errors. Set to [] to ignore check for positive definiteness. *Default:* `[]`
- `'embedDim'`: Dimensionality of spectral embedding to use in spectral clustering, which is typically the same as the number of classes. Default: `k`
- `'emptyVal'`: Block density value to set for empty blocks, which occur when a class has no nodes. *Default:* `0.001`
- `'innovCov'`: Covariance matrix of innovation $\mathbf{e}^t = \mathbf{y}^t - h\left(\widehat{\boldsymbol{\psi}}^{t|t-1}\right)$. This can be specified to save computation in the a posteriori setting, because the innovation does not change with respect to the class membership estimates in the local search. Set to `[]` to compute innovation covariance using the EKF. *Default:* `[]`
- `'kalmanGain'`: Kalman gain matrix. Similar to innovCov, this can be specified to save computation. Set to `[]` to compute Kalman gain using the EKF. *Default:* `[]`
- `'maxIter'`: Maximum number of iterations of local search to perform. *Default:* `100`
- `'model'`: Model index vector for state transition model and state transition covariance matrices $F^{1:t}$ and $\Gamma^{1:t}$, respectively. `model(t) = m` denotes that the mth slice of the state transition model and covariance matrices should be used at time t. *Default:* all ones vector
- `'nClasses'`: Maximum number of classes. It usually does not need to be manually specified, but is available to set the maximum number of classes larger than the number of classes in the estimated class membership vector. *Default:* `max(class)`
- `'nKmeansReps'`: Number of replicates (random initializations) of k-means clustering to use in spectral clustering. *Default:* `1`
- `'nPairs'`: Number of pairs of nodes (possible edges) in each block. This is used to estimate the covariance matrix Σ^t of the block densities, if Σ^t is not already specified (set to `[]`). If Σ^t is specified, this input is not used. *Default:* `[]`
- `'obsClip'`: How much to clip the range of block densities away from the boundaries of 0 and 1, for which the logit is undefined. *Default:* `0.001`

- `'svdType'`: Type of singular value decomposition to use for spectral clustering, either `'full'` or `'sparse'`. `'sparse'` is recommended for large networks (more than a few hundred nodes). *Default:* `'full'`
- `'output'`: Level of output to print to command window. Higher values result in more output, and `0` results in no output. *Default:* `0`
- `'stateClip'`: Magnitude to clip state estimates. Since states are logits of edge probabilities, they should not deviate too far from 0 so that edge probabilities are not too close to 0 or 1. *Default:* `10`

Recommended toolboxes

- Parallel Computing Toolbox: required to conduct local search over SBM classes in parallel for a posteriori blockmodeling. This toolbox is highly recommended to speed up the computation time for a posteriori blockmodel inference.
- Statistics Toolbox: required to use spectral clustering to estimate the initial class estimates. If the Statistics Toolbox is not available, substitute an alternate function for k-means clustering in `spectralClusterSbm.m`.

Description of files

- `blockmat2vec.m`: Convert $k \times k$ matrix of representation of state or observation into p-dimensional vector representation (see State representation section for details).
- `blockvec2mat.m`: Convert p-dimensional vector representation of state or observation into $k \times k$ matrix representation (see State representation section for details).
- `calcBlockDens.m`: Calculate block densities given time series of adjacency matrices and class memberships.
- `ekfDsbm.m`: Fit dynamic stochastic blockmodel in a priori setting with specified classes using extended Kalman filter.
- `ekfDsbmLocalSearch.m`: Fit dynamic stochastic blockmodel in a posteriori setting using extended Kalman filter and local search to estimate class memberships.
- `ekfPredict.m`: Prediction phase of extended Kalman filter.
- `ekfUpdate.m`: Update (correction) phase of extended Kalman filter.
- `estimateDsbmProb.m`: Estimate edge probability matrix for dynamic stochastic blockmodel at time t given adjacency matrix, class membership vector, and extended Kalman filter prediction.
- `estimateSbmProb.m`: Estimate edge probability matrix for static stochastic blockmodel given adjacency matrix and class membership vector.
- `generateDsbm.m`: Generate time series of adjacency matrices according to a dynamic stochastic blockmodel.
- `generateSbm.m`: Generate an adjacency matrix according to a static stochastic block model.
- `generateStateCov.m`: Generate state covariance matrix such that all state variances are identical, and all states corresponding to neighboring blocks have the same covariance.

- `localSearchDsbm.m`: Perform local search to estimate class memberships at a given time step for a dynamic stochastic blockmodel.
- `localSearchSbm.m`: Perform local search to estimate class memberships for a static stochastic blockmodel.
- `permuteClasses.m`: Permute a class membership vector for maximum agreement with a reference class membership vector.
- `predAdjMatDsbm.m`: Forecast adjacency matrix at next time step using dynamic stochastic blockmodel states.
- `spectralClusterSbm.m`: Perform spectral clustering on adjacency matrix to estimate class membership vector for a static stochastic blockmodel.

Changelog

Version 0.2

- Updated Readme with guidance on choosing hyperparameters
- Fixed bug in a posteriori local search procedure that prevented local search from starting
- Fixed bug in clipping block densities

Version 0.1

- Initial release containing a priori and a posteriori inference algorithms with two data experiments: a simulated experiment and an experiment on the Enron email corpus

Appendix: Description of additional files

Most users should not need to interact with the following files. Unless otherwise noted, all files were written by the authors of this toolbox.

- `adjmat2vec.m`: Convert adjacency matrix to a vector representation. Used to evaluate link forecasting.
- `adjvec2mat.m`: Convert vector representation of adjacency matrix back to matrix representation.
- `bernrnd.m`: Generate random samples from a Bernoulli distribution.
- `cluvec2mat.m`: Convert class membership vector to binary indicator matrix form.
- `clumat2vec.m`: Convert binary class indicator matrix to class membership vector.
- `setDefaultParam.m`: Used to set default values for optional parameters in functions.

`valid_RandIndex.m` is redistributed without modification from the MATLAB toolbox for estimating the number of clusters (Wang, 2009).

References

Priebe, C. E., Conroy, J. M., Marchette, D. J., & Park, Y. (2009). Scan statistics on Enron graphs. Retrieved from http://cis.jhu.edu/~parky/Enron/enron.html

Wang, K. (2009). (Simple) Tool for estimating the number of clusters. Retrieved from http://www.mathworks.com/matlabcentral/fileexchange/13916--simple--tool-for-estimating-the-number-of-clusters

Xu, K. S., & Hero III, A. O. (2014). Dynamic stochastic blockmodels for time-evolving social networks. IEEE Journal of Selected Topics in Signal Processing, 8(4), 552–562. doi:10.1109/JSTSP.2014.2310294