Non-stationary Dynamic Bayesian Networks

Joshua W. Robinson, Alexander J. Hartemink
Department of Computer Science, Duke University
{josh,amink}@cs.duke.edu

Abstract: Structure learning of dynamic Bayesian networks provide a principled mechanism for identifying conditional dependencies in time-series data. This learning procedure assumes that the data are generated by a stationary process. However, there are interesting and important circumstances where that assumption will not hold and potential non-stationarity cannot be ignored. Here we introduce a new class of graphical models called non-stationary dynamic Bayesian networks, in which the conditional dependence structure of the underlying data-generation process is permitted to change or evolve over time. Some examples of evolving networks are transcriptional regulatory networks during development, neural pathways during learning, and traffic patterns during the day. We define the non-stationary DBN model, present an MCMC sampling algorithm for efficiently learning the structure of an nsDBN and the times of non-stationarities (transition times) under different assumptions, and demonstrate the effectiveness of the algorithm on simulated data.

1 Previous Work

In recent work from the social networks community, a generalization of the p^* or exponential random graph model (ERGM) to account for temporal dynamics has been used to model the temporal progression of networks [2]. However, this algorithm requires the network structures as input, a requirement which limits usability in other fields. This approach was recently extended by assuming that the networks are latent (unobserved) variables which generate the observed time-series data [1]. While this technique allows for the underlying network structure to be identified, there are some drawbacks: the correlation structure between variables is assumed to remain constant over time, only undirected edges are predicted, and the transition times must be identified *a priori*.

In the continuous domain, there has been some research focused on learning the structure of a time-varying Gaussian graphical model [4]. These authors use a reversible-jump MCMC to estimate the time-varying variance structure of the data. However, some limitations of this method include: network evolution is restricted to changing at most a single edge at a time and the total number of segments is assumed known *a priori*. A similar algorithm using Gaussian graphical models has been developed to segment multivariate time-series data [5]. This work uses an iterative approach that switches between a convex optimization for determining the graph structure and a dynamic programming algorithm for calculating the segmentation. This approach has some notable advantages: speed, no single edge change restriction, and number of segments calculated *a posteriori*; however, it requires that the graph structure is decomposable. Additionally, both of these approaches only identify undirected edges and assume that the networks in each segment are independent. By extending Bayesian networks, we are able to make directed predictions and we allow the networks in one segment to depend on those in adjacent segments.

2 Algorithmic Details

Assume that we observe the state of n random variables at N discrete times indexed $0, 1, \ldots, N-1$. Call this multivariate time-series D. We call the initial network of conditional dependencies G_1 and subsequent networks are called G_i for $i = 2, 3, \ldots, m$. We define Δg_i to be the set of edges that change (either added or deleted) between G_i and G_{i+1}. We define the *transition time* t_i to be the time at which G_i is replaced by G_{i+1} in the data-generation process. The set of transition times is $T = \{t_1, \ldots, t_{m-1}\}$. We call the period of time between consecutive transition times—during which a single network of conditional dependencies is operative—an *epoch*. So we say that G_1 prevails during the first epoch, G_2 prevails during the second epoch, and so forth. We will refer to the entire series of prevailing networks as the *structure* of the nsDBN.

If we know all of the transition times *a priori*, then we need to identify a structure that maximizes $P(G_1, \Delta g_1, \ldots, \Delta g_{m-1} | D, T)$. Instead, we can maximize the probability of the data given the nsDBN structure using Bayes rule. As with most structure learning techniques, the best structure can be identified via a heuristic search or a sampling strategy. We decided to use an MCMC sampling approach similar to that described in [3] because a sampling algorithm of this form can be easily extended for the inference of nsDBNs in settings with increasing levels of uncertainty.

For a sampling strategy to work effectively, one needs to consider which local moves result in jumps between posterior modes. Networks that differ by a single edge will probably have similar likelihoods. Therefore, we include in the move set a single edge addition or deletion to G_1. This change is indirectly propagated through the other networks in the structure because they each differ from the previous network by the edge set Δg_i. Additionally, we consider adding or deleting an edge in a particular Δg_i and moving an edge from one Δg_i to another. The resultant move set results in adequate mixing behavior when we know the transition times but not the actual edge changes.

Knowing in advance the times at which all of the transitions occur is often unrealistic. In this setting, we need to provide some initial transition times and then shift their locations to maximize the posterior. This can be accomplished by augmenting the

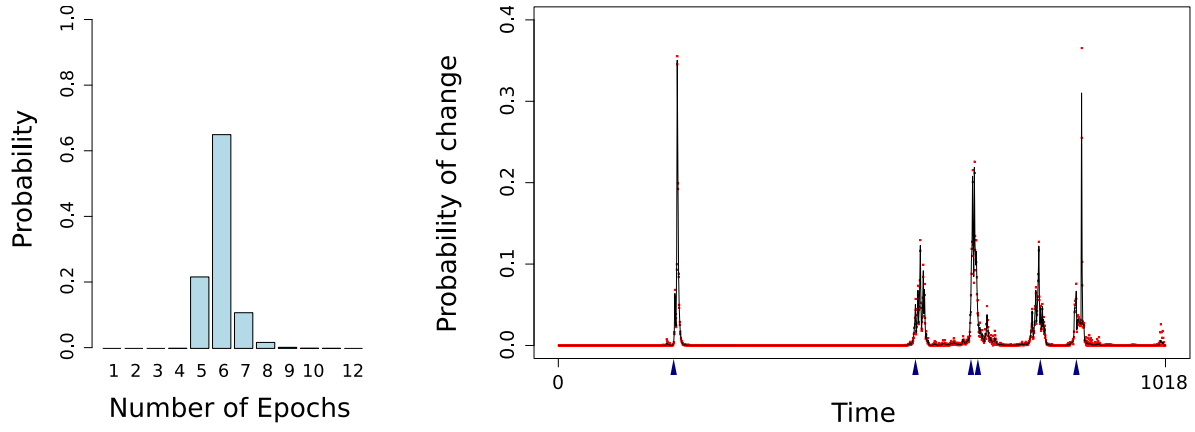

Figure 1: Results on simulated data for a particular setting of λ_m and λ_e. *Left*: Posterior probabilities of the total number of epochs (the true number is seven). *Right*: Posterior probability of transition times when learning an nsDBN when neither the number of transition of the transition times are known. The blue triangles represent the true transition times and the red dots represent one standard deviation from the mean probability obtained from several runs.

move set to include local shifts to t_i. Interestingly, the addition of this option to the move set does not appreciably slow the convergence rate.

The most general setting is when both the transition times T and the number of transitions m are unknown and have to be estimated. While this is the most interesting setting, it is also the most difficult. We again augment the move set to allow for transitions to be added or removed. To allow the number of transitions to change, we introduce merge and split operations to the move set. For the merge operation, two adjacent edge sets (Δg_i and Δg_{i+1}) are combined to create a new edge set. For the split operation, an edge set Δg_i is randomly chosen and randomly partitioned into two new edge sets $\Delta g_i'$ and Δg_{i+1}. Finally, we include the add and delete transition time operations which add a random edge change at a random transition time and delete all the edge changes at a single transition time, respectively.

To prevent overfitting (i.e., a transition predicted to occur at every observation), regularization terms also need to be added. Several aspects of the network may be regularized; we currently regularize based on the total number of edge changes e and the number of epochs m. If an exponential prior with rate λ_e is placed on the total number of edge changes, and one with a rate of λ_m is placed on the total number of epochs, then the updated likelihood for the structure is given as:

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T|D)\frac{1}{Z}\mathrm{e}^{-\lambda_e e}\mathrm{e}^{-\lambda_m m}$$

where Z is a normalization constant. Although we have chosen to place an exponential prior on the number of epochs, note that this is equivalent to placing a geometric prior on the epoch lengths, as the authors did in [4].

We have tested this method for each of the above settings on both simulated and real data. Shown in Figure 1 are the posterior number of epochs and the posterior probabilities of transition time locations for a ten variable simulated network. Both the predicted number of epochs and the most probable transition time locations are very close to the truth. In other simulations, we tested the scalability of the algorithm by increasing the number of nodes to 100. While it took longer to converge, the total running time was still within reasonable bounds, and the predictions were just as accurate.

3 Conclusion

Non-stationary dynamic Bayesian networks provide a useful framework for learning Bayesian networks when the generating processes are non-stationary. Using the move sets described here, the task of learning nsDBNs is reasonably efficient and is generalizable to situations of varying uncertainty. An nsDBN can be learned from any type of data with dynamic behavior that can be described as a network, such as traffic utilization on roads or neural networks of information flow. We hope that this model will provide an attractive alternative to researchers who have, in the past, resorted to segmenting the data and learning separate networks.

References

[1] F. Guo, S. Hanneke, W. Fu, and E. P. Xing. Recovering temporally rewiring networks: A model-based approach. In *ICML07*, 2007.

[2] S. Hanneke and E. P. Xing. Discrete temporal models of social networks. In *Workshop on Statistical Network Analysis, ICML06*, 2006.

[3] D. Madigan, J. York, and D. Allard. Bayesian graphical models for discrete data. *Intl. Statistical Review*, 63(2):215–232, 1995.

[4] M. Talih and N. Hengartner. Structural learning with time-varying components: Tracking the cross-section of financial time series. *J. Royal Stat. Soc. B*, 67(3):321–341, 2005.

[5] X. Xuan and K. Murphy. Modeling changing dependency structure in multivariate time series. In *ICML07*, 2007.