Learning Time-Varying Graphs using Temporally Smoothed L_1-Regularized Logistic Regression

Amr Ahmed **Eric P. Xing**

September 2008
CMU-ML-08-119

MACHINE LEARNING
DEPARTMENT

Carnegie Mellon

Learning Time-Varying Graphs using Temporally Smoothed L_1-Regularized Logistic Regression

Amr Ahmed **Eric P. Xing**

September 2008
CMU-ML-08-119

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A plausible representation of the relational information among entities in dynamic systems such as a living cell or a social community is a stochastic network which is topologically rewiring and semantically evolving over time. While there is a rich literature on modeling static or temporally invariant networks, until recently, little has been done toward modeling the dynamic processes underlying rewiring networks, and on recovering such networks when they are not observable. In this paper we present an optimization-based approach for recovering time-evolving discrete networks from time stamped node samples from the network. We cast this graphical model learning problem as a temporally smoothed L1-regularized logistic regression problem which can be formulated and solved efficiently using standard convex-optimization solvers scalable to large networks. We report promising results on recovering the dynamics of the coauthorship-keyword academic social network in the NIPS conference.

Keywords: Time-Varying networks, optimization, Lasso, logistic regression

1 Introduction

In many problems arising in social, biological, and other fields, it is often necessary to analyze populations of individuals (actors), interconnected by a set of relationships (e.g., friendship, communication, influence, etc.) represented as a **network**. Real-time analysis of network data is important for detecting anomaly, predicting vulnerability, and assessing the potential impact of interventions in various social, biological, or engineering systems. It is not unusual for network data to be large, dynamic, heterogeneous, noisy and incomplete. Each of these characteristics adds a degree of complexity to the interpretation and analysis of networks.

While there is a rich (and growing) literature on modeling a static network at a single point in time, or time-invariant networks [19], with few exceptions [1,14,17,7], much less has been done toward modeling the dynamical processes underlying networks that are topologically rewiring and semantically evolving over time; and on developing efficient learning techniques, especially in a dynamic context, for recovering unobserved network topologies from observed attributes of entities constituting the network. Recently, a new class of models known as the *temporal Exponentially Random Graph Model* (tERGM) has been proposed for modeling networks evolving over discrete time steps [14]. It is based on a log-linear graph transition model defined on a set of flexibly designed *temporal potentials* that capture certain characteristics of the graph rewiring dynamics, such as the "edge-stability", "density", and "transitivity" statistics over time-adjacent graphs. Based on this model, subsequently a *hidden* tERGM model was proposed so that given time series of node attributes, one can infer time-specific network topology based on the posterior distribution of the hidden network given the node attributes over the entire time series [1]. As expressive as it sounds, and as discussed by the authors in [1], the Gibbs sampling algorithm for posterior inference under this model is very inefficient, and can scale only to few tens of nodes because of the presence of the complex partition functions of the Boltzmann-distribution based graph transition model and graph emission models; and up till now, providing an efficient inference and learning algorithms for this model is still an open problem. The work we present here focuses on modeling graphs that evolve smoothly over time. Along these lines, very recently, [17] proposed non-parametric local-kernel smoothing techniques for learning smoothly evolving undirected Gaussian graphical models using the method in [15], and showed interesting asymptotic consistency results of their estimators. In contrast, here we address the case of *discrete* time-evolving graphs.

In this paper, we propose a new approach [1] for recovering time-evolving networks on *fixed* set of nodes from time series of entity attributes using *temporally smoothed* L_1*-regularized logistic regression*, or in short, TLR (for temporal LR). The time series node attributes are assumed to be discrete; and for concreteness, we will focus on the binary case which covers the class for learning time-evolving pairwise Markov random field networks. The TLR can be formulated and solved using existing efficient convex optimization techniques which makes it readily scalable to learning evolving graphs with hundreds and *potentially* few thousands of nodes. The rest of this paper is organized as follows. In Section 2 we begin with a basic formulation of the graph learning problem. Section 3 then discusses and contrasts various smoothness penalties. Section 4 details the TLR method. In Section 5 we show experimental results, and we conclude in Section 6.

[1] A very preliminary version of this work appeared in [21].

2 Problem Formulation

Let $G^t = (V, E^t)$ be the graph structure at time epoch t with vertex set V of size $|V|$ = p and edge set E^t. Let $\{X^t_{1:N_t}\}$ be a set of i.i.d *binary* random variables associated with the vertices of the graph. Let the joint probability of the random variables be given by the Ising model as follows:

$$P(\mathbf{x_d^t}|\Theta^t) \;=\; \exp\left(\sum_{i\in V}\theta^t_{ii}x^t_{d,i} + \sum_{(i,j)\in E^t}\theta^t_{ij}x^t_{d,i}x^t_{d,j} - A(\Theta^t)\right), \tag{1}$$

where the parameters $\{\theta^t_{ij}\}_{(i,j)\in E^t}$ capture the correlation between the variables X^t_i, and X^t_j. $A(\Theta^t)$ is the log normalizing constant of the distribution. Given a set of $\{X^t_{1:N_t}\}$ i.i.d samples drawn from $P(X^t_d|\Theta^t)$ at each time step, the goal is to estimate the structure of the graph, i.e. to estimate $\{\hat{E}^t\}^T_{t=1}$. We cast this problem as a regularized estimation problem of the time-varying parameters of the graph as follows:

$$\hat{\Theta^1}, \ldots, \hat{\Theta^T} \;=\; \arg\min_{\Theta^1,\ldots,\Theta^T} \;\sum_{t=1}^{T} nLL(\Theta^t) \;+\; R(\Theta^1 \ldots \Theta^T, \lambda), \tag{2}$$

where, $nLL(\cdot)$ is the exact (or approximate surrogate) of the negative Log Likelihood $R(\cdot)$ is a regularization term, and λ is the regularization parameter(s). We assume that the graph is *sparse* and evolves *smoothly* over time, and we would like to pick a regularization function $R(\cdot)$ that results in a sparse and smooth graphs. Since we are restricting our attention on binary pairwise MRF, the structure of the graphs can then be recovered from the non-zero parameters which are isomorphic to the edge set of the graphs. In the next section we will first examine various options for $R(\cdot)$, then in section 4, we will present our approach.

3 Smoothness, Sparseness and the L_1-Penalty

Regularization of optimization problems using a penalty function is a standard approach in machine learning. This is done either to improve generalization or to encode prior information about the problem into the model. Recent research has found that regularizing an optimization problem by an L_1-norm penalty of the parameter vector leads to sparse solutions [8,10]. The L_1-norm of a P-dimensional vector θ, is defined as $||\theta||_1 = \sum_{i=1}^{P}|\theta_i|$. Compared to the standard L_2^2 norm, defined as $||\theta||_2^2 = \sum_i \theta_i^2$, for small values of $\theta < 1$, $||\theta||_1 << ||\theta||_2^2$. Therefore, if one were to minimize the l_1-norm of a vector, it drives many of the small components to zeros, unlike the square of the L_2-norm penalty, which is more tolerating to small residuals [8]. Hence, one can enforce sparsity of a model in an optimization problem by using a regularization term that minimizes the L_1-norm of the parameter vector. This penalty has been used extensively in the literature (see [13] for a review) to various loss functions: least square loss (linear regression) [10], logistic regression [3,2],etc. Historically, when the L_1-penalty is applied to least square linear regression, it derives the famous name, the $lasso$ [10].

In many learning problem, there are correlation between the variables, and thus one might expect that correlated variables should be assigned similar parameter values. While the L_1-penalty

can successfully enforces sparseness over the parameter vector, it can not do block selection/removal of a group of correlated variables simultaneously. Several extensions have been proposed in the literature to extend the L_1-penalty to enforce smoothness in addition to sparsity over the parameter vector. These extensions differ in the way correlation between the variables is represented. In [9] the authors proposed the fused-lasso penalty, which we will refer to as $L_1 + L_1$ in this paper, which takes the following form:

$$L_1 + L_1(\theta, \lambda_1, \lambda_2) \quad = \quad \lambda_1 \sum_{i=1}^{P} |\theta_i| + \lambda_2 \sum_{i=2}^{P} |\theta_i - \theta_{i-1}| \tag{3}$$

In this penalty the order of the variables is used as a proxy for correlation and is supplied using domain knowledge to the model. This penalty has been shown in [9] to result in sequences of zero (or non-zero) parameters thus enforcing smoothness and sparsity. In [11] the authors proposed an explicit representation of the correlation structure between the variables in terms of groups (or blocks) and then the L_1-penalty is applied to a summary of the parameters in this block, hence came the now the block(group)-regularization. For instance, an L_∞ norm can be used over the parameters in each block (thus selecting the maximum parameter in each block) and then an L_1-penalty is applied to the resulting block maximums (although several other combinations are possible). More recently [20] proposed network-regularization in which the correlation between the variables is encoded via edges in a correlation graph. The parameter values are then regularized using the graph Laplacian, L, which forces neighboring variables to be assigned smooth parameters: $R_{\text{network}}(\theta, \lambda_1, \lambda_2) = \lambda_1 \sum_{i=1}^{P} |\theta_i| + \lambda_2 \theta^T L \theta$. It is interesting to note that when the graph structure degenerates to a linear chain, up to a constant, the network regularization will have a form similar to the fused lasso, which we will refer to in this paper as $L_1 + L_2^2$:

$$L_1 + L_2^2(\theta, \lambda_1, \lambda_2) \quad = \quad \lambda_1 \sum_{i=1}^{P} |\theta_i| + \lambda_2 \sum_{i=2}^{P} (\theta_i - \theta_{i-1})^2 \tag{4}$$

To understand the tension between all these forms, in Figure 1, we draw the contour levels of four regularization penalties: L_1, L_1+L_1,$L_1+L_2^2$ and L_2^2 over two parameters β_1 and β_2. First,as we mentioned above, the L_1 is sharper than the L_2^2 for small values, and thus has the ability to enforce sparsity. Second, both the $L_1 + L_1$ and $L_1 + L_2^2$ penalties are not symmetric around the horizontal and vertical axes, which is in fact a desirable property. For example, the parameter vector (1,-1.1) is penalized differently than its x-mirror image (-1,-1.1), as the first is not smooth while the later is. However, the L_1 and L_2^2 penalties are oblivious to this relationship. Moreover, the smoothed penalties are symmetric around the 45-degrees tilted axes, therefore, parameter vectors (1,1.1) and (-1,-1.1) are penalized similarly. However,the $L_1 + L_1$ differs from the $L_1 + L_2^2$ in the same way the L_1 differs from L_2: the former is sharper for small differences of parameter vectors than the later as shown in Figure 1 via the sharp vs. smooth curvatures around the axes of symmetry.

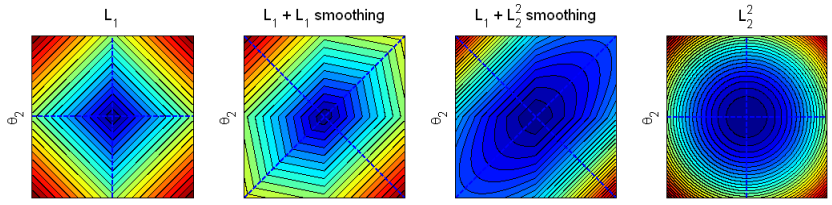

Figure 1: An illustration of various penalty functions. The dotted lines, denote the axes of symmetry, see section3 for more details.

4 Temporally Smoothed L_1-Regularized Logistic Regression for Learning Evolving Graphs

Now we detail our method for learning smoothly temporally evolving sparse graphs. Our goal, as stated in problem casted in Eq. (2), is learning *sparse* and *smooth* edge weights $\{\theta_{ij}^t\}$ for all i, j, and t. We begin with the static case to review how L_1-regularized logistic regression is used in structure learning, then we present our temporally smoothed L_1-regularized logistic regression approach in section 4.2 for the dynamic setting.

4.1 Learning Static Discrete Graphs

When $T = 1$, the problem in Eq. (2) degenerates to the static case:

$$\hat{\Theta} \quad = \quad \arg\min_{\Theta} \; nLL(\Theta) + R(\Theta, \lambda), \tag{5}$$

and thus one needs only to use a regularization function, $R(\cdot)$, that enforces sparsity. In [2,5,15] this problem has been addressed by choosing $R = L_1$-penalty, however, they differ in the way they *approximate* the first term in Eq. (5) which is intractable in general due to the existence of the log partition function, $A(\Theta)$. In [5], belief propagation was used to approximate $A(\Theta)$ for gradient computations; in [15] a log-determinant relaxation was employed; whereas in [2] a pseudo-likelihood was used. Our work builds on [2] thus we will illustrate their method in more details. The pseudo-likelihood, $\hat{P}(X_d|\Theta) = \prod_{i=1}^{P} P\big(x_{d,i}|\mathbf{x_{d,N(i)}}\big)$, where $N(i)$ is the Markov blanket of node i, i.e., the neighboring nodes of node i. In the binary pairwise-MRF, this local likelihood has a logistic-regression form. Thus the learning problem in (5) degenerates to solving P l_1-regularized logistic regression problems resulting from regressing each individual variable on all the other variables in the graph. More specifically, the learning problem for node i is given by:

$$\begin{aligned}
\hat{\theta}_i \quad &= \quad \arg\min_{\theta_i} \frac{1}{N} \sum_{d=1}^{N} \log \; P\big(x_{d,i}|\mathbf{x_{d,-i}}, \theta_i\big) + \lambda_1 \parallel \theta_{-i} \parallel_1 \\
&= \quad \arg\min_{\theta_i} \frac{1}{N} \sum_{d=1}^{N} \Big[\log(1 + \exp(\theta_{\mathbf{i}} \mathbf{x_{d,-i}})) - x_{d,i} \theta_{\mathbf{i}} \mathbf{x_{d,-i}}\Big] + \lambda_1 \parallel \theta_{\mathbf{-i}} \parallel_1,
\end{aligned} \tag{6}$$

where, $\theta_{\mathbf{i}} = (\theta_{i1}, \ldots, \theta_{iP})$ are the parameters of the L_1-logistic regression, $\mathbf{x_{d,-i}}$ denotes the set of all variables with $x_{d,i}$ replaced by 1, and θ_{-i} denotes the vector θ_i with the component θ_{ii} removed (i.e. the intercept is not penalized). The estimated set of neighbors is given by: $\hat{N}(i) = \{j : \theta_{ij} \neq 0\}$. The set of edges E is then defined as either a union or an intersection of neighborhood sets $\{N(i)\}_{i \in V}$ of all the vertices. Wainwright et al [2] showed that both definitions would converge to the true structure asymptotically.

4.2 Learning Time-Varying Discrete Graphs

Now we return to the original learning problem in Eq. (2) and approach it using the techniques presented above. We use the negative pseudo-loglikelihood as a surrogate for the intractable $nLL(\cdot)$ at each time epoch. To constrain the multiple time-specific regression problems each of which takes the form of Eq. (6) so that graphs are evolving in a smooth fashion, that is, not dramatically rewiring over time, we penalize the difference between the regression coefficient vectors corresponding to the same node, say i, at the two adjacent time steps. This can be done by introducing a regularization term $\parallel \theta_i^t - \theta_i^{t-1} \parallel_q^q$ for each node at each time. It is noteworthy that this term can be regarded as the "edge-stability" potential in the htERGM model [1]. Then, to also enforce sparsity over the graphs learnt at each epoch, in addition to the smoothness between their evolution across epochs, we use the standard L_1 penalty over each θ_i^t. Putting things together, now we have the regularization function $R(\cdot)$ as either the $L_1 + L_1$ penalty in Eq. (3), or the $L_1 + L_2^2$ penalty in Eq. (4). These choices will decouple the learning problem in (2) into a set of P separate *smoothed* L_1-regularized logistic regression problems, one for each variable. Putting everything together, for each node i in the graph, we solve the following problem for $q = 1$ (R =$L_1 + L_1$) or $q = 2$ (R = $L1 + L_2^2$):

$$\hat{\theta}_i^1, \ldots, \hat{\theta}_i^T \;=\; \arg\min_{\theta_i^1, \ldots, \theta_i^T} \sum_{t=1}^{T} l_{avg}(\theta_i^t) + \lambda_1 \sum_{t=1}^{T} \parallel \theta_{-i}^t \parallel_1 + \lambda_2 \sum_{t=2}^{T} \parallel \theta_i^t - \theta_i^{t-1} \parallel_q^q, \tag{7}$$

where $l_{avg}(\theta_{\mathbf{i}}^{\mathbf{t}}) = \frac{1}{N^t} \sum_{d=1}^{N^t} \log\, P\big(x_{d,i}^t | \mathbf{x}_{\mathbf{d},-\mathbf{i}}^{\mathbf{t}}, \theta_{\mathbf{i}}^{\mathbf{t}}\big) = \frac{1}{N^t} \sum_{d=1}^{N^t} \big[\log(1 + \exp(\theta_{\mathbf{i}}^{\mathbf{t}} \mathbf{x}_{\mathbf{d},-\mathbf{i}}^{\mathbf{t}})) - x_{d,i}^t \theta_{\mathbf{i}}^{\mathbf{t}} \mathbf{x}_{\mathbf{d},-\mathbf{i}}^{\mathbf{t}}\big]$.

The problem in Eq. (7) is a convex optimization problem with a non-smooth L_1 function. When $q = 1$, we solve the following equivalent problem instead by introducing new auxiliary variables, $\mathbf{u_i^t}$ and $\mathbf{v_i^t}$ (the case for $q = 2$ is handled similarly):

$$\begin{aligned} \min_{\theta_i^1, \ldots, \theta_i^T, u_i^1, \ldots, u_i^T, v_i^2, \ldots, v_i^T} & \sum_{t=1}^{T} l_{avg}(\theta_i^t) + \lambda_1 \sum_{t=1}^{T} \mathbf{1}^T \mathbf{u_i^t} + \lambda_2 \sum_{t=2}^{T} \mathbf{1}^T \mathbf{v_i^t} \\ \text{subject to} \quad & -u_{i,j}^t \leq \theta_{i,j}^t \leq u_{i,j}^t,\ t = 1, \ldots, T, j = 1, \ldots, i-1, i+1, \ldots, P, \\ \text{subject to} \quad & -v_{i,j}^t \leq \theta_{i,j}^t - \theta_{i,j}^{t-1} \leq v_{i,j}^t,\ t = 2, \ldots, T, j = 1, \ldots, P, \end{aligned} \tag{8}$$

where $\mathbf{1}$ denotes a vector with all components set to 1, so $\mathbf{1}^T \mathbf{u_i^t}$ is the sum of the components of $\mathbf{u_i^t}$. To see the equivalence of the problem in Eq. (8) with the one in Eq. (7) for $q = 1$, we note that at the optimal point of Eq. (8), we must have $u_{i,j}^t = |\theta_{i,j}^t|$, and similarly $v_{i,j}^t = |\theta_{i,j}^t - \theta_{i,j}^{t-1}|$, in which case the objectives in Eq. (8) and Eq. (7) are the same (a similar solution has been

applied to solving L_1-regularized logistic regression in [3]). The problem in Eq. (8) is a convex optimization problem, with now a smooth objective, and linear constraint functions, so it can be solved by standard convex optimization methods, such as interior point methods, and high quality solvers can directly handle the problem in Eq. (8) efficiently for small and medium scale (up to few hundred nodes and few hundred samples). In this paper, we used the CVX optimization package [16].

4.2.1 Hyperparmaters Selection

The hyperparmaters in Eq. (7) trades off spareness, smoothness versus locally fitting the time-epoch specific samples. Larger values of λ_2 degenerates the problem into the static case in which all the time-specific parameters are equal to each other, while setting λ_2 to zero decouples Eq. (7) into a set of independent T L_1-regularized logistic regression problems, one at each epoch. On the other other hand, λ_1 controls the spareness of the resulting graphs. To tune these parameters one first should note that the problem in Eq. (7) is solved independently for every node, thus in practice one can tune these parameters independently for every node. Moreover, we note also that Eq. (7) can be regarded as a *supervised* classification problem, and thus many techniques can be used to select (λ_1, λ_2) among different candidates. When there are enough data, cross-validation, or held-out datasets can be used, otherwise, the BIC score can be employed. We define the BIC score for $(\theta_i^1, \ldots, \theta_i^T)$ to be:

$$BIC(\theta_i^1, \ldots, \theta_i^T) \approx \sum_{t=1}^{T} -l_{avg}(\theta_i^t) - \frac{\log(\sum_{t=1}^{T} N_t)}{2} \mathrm{Dim}(\theta_i^1, \ldots, \theta_i^T), \quad (9)$$

where $Dim(\cdot)$ denotes the dimensionality of the estimated values. Similar to [9], we adopt the following definition, which counts the number of runs of non-zero parameter values.

$$Dim(\theta_i^1, \ldots, \theta_i^T) = \sum_{t=1}^{T} \sum_{j=1}^{P} I\left[sign\left(\theta_{i,j}^t\right) \neq sign\left(\theta_{i,j}^{t-1}\right)\right] * I\left[sign(\theta_{i,j}^t) \neq 0\right]. \quad (10)$$

Note that the above definition is lenient to perturbations to the value of the actual parameters, whereas, the definition in [9] strictly increases the dimensionality of the model with these perturbations. Our intuition here is that a non-zero parameters corresponds to an edge, thus the above measure counts the number of edge changing events (including change in polarity).

5 Academic Author-keyword Social Network

We applied our approach using $L_1 + L_1$ on the NIPS12 dataset[2], which contains the proceeding of the NIPS conference from years 1987-1999, to analyze the dynamics of the academic social network between authors and keywords. We selected the top 36 authors based on publication counts, which resulted in a total of 431 papers. We then selected relevant 73 words from these

[2]Available from http://www.cs.toronto.edu/ roweis/data.html

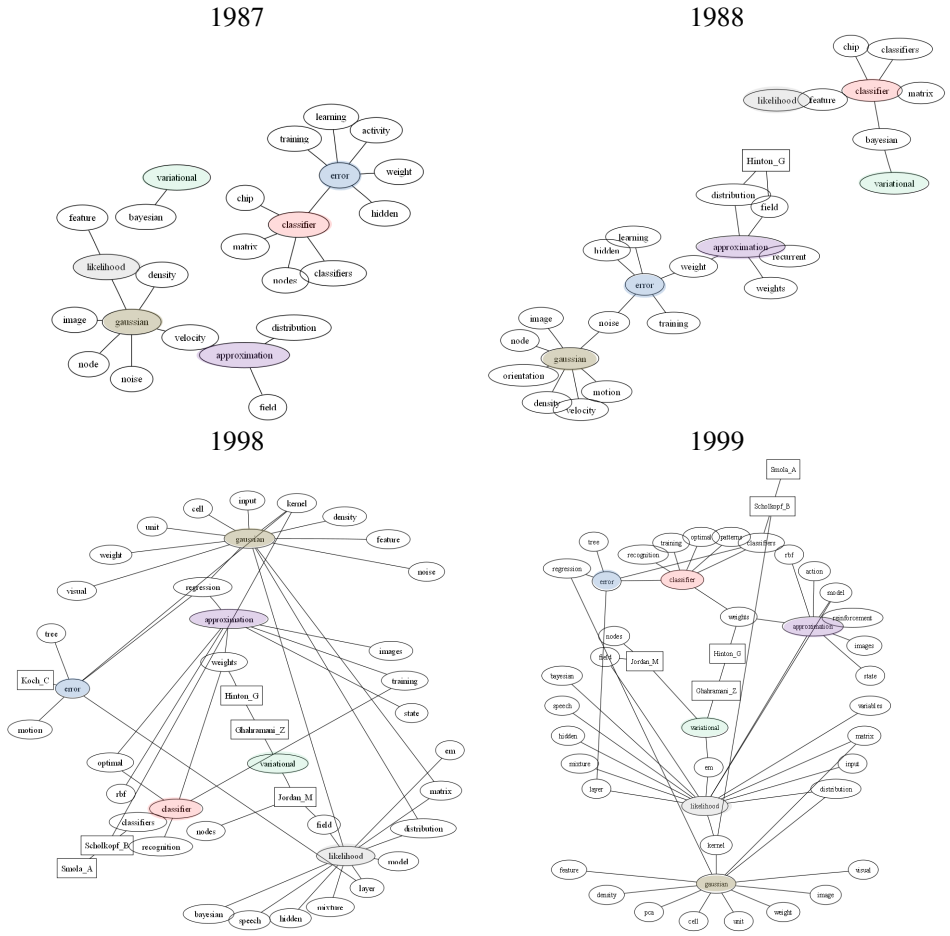

Figure 2: Illustrating the dynamics of NIPS academic social network. Tracked words are highlighted, and authors are drawn in rectangles. See section 5 for more details.

papers and binarized the results. Thus each sample from this graph has dimensionality 109. We then divided the data across 13 epochs using the time stamp of the paper. On a standard desktop running 2.4GHZ processor with 2GB memory, the proposed method,using the CVX generic optimization package [16], took around 2.5 hours for all the nodes combined (an average of 50-80 seconds per node). To illustrate the result, we selected 6 keywords:'gaussian','classifier' , 'likelihood','approximation','error' and 'variational' (highlighted in Figure 2) and tracked their subnetworks up to the first neighbors due to space limitations. Authors are shown in the same graph using rectangles. In figure 2 we show the first and last two epochs. Before interpreting this result, one need to recall the semantic of neighbors in a MRF: knowing the values of all the variables in the Markov Blanket (MB) of a given node, the prediction we made about the value of this node will remain the same no matter how many other nodes are observed. Thus one would expect that words which appears in many contexts would have a larger MB than very specialized words. Looking back at Figure 2, first, one should observe the smooth transition between the networks from 1987 to 1988 and from 1998 to 1999. Second, as discussed above, the MB of a word is a good indication of the diversity of this word usage. For instances, in early years, the word 'likelihood'

had a limited context in 'gaussian' settings, however, over the years, this behavior changed and at the year 1998 and 1999 the word 'likelihood' started to appear in different contexts like 'speech', 'bayesian', 'mixture', etc. and the model expanded its MB over the years. On the other hand, words like 'Gaussian' were always popular in different settings, and gets more popular over the years and as such the model smoothly expanded its MB as well. It is also interesting to analyze the word 'variational', where in early years it was tightly coupled with the word 'Bayesian', and over the years the MB of the word 'variational' was dominated by 'Jordan' and 'Ghahramani' to the point that during the year 1998, the model believed that it is quite enough to know if 'Jordan' and 'Ghahramani' were mentioned in a paper to decide if the word 'variational' will appear or not independently of any other observations. Then over the year 1999, the MB of the word 'variational' expanded to include the word 'EM' as in 'variational EM'. Also, note how 'Hinton' was always connected to terms related to Boltzmann machines like 'distribution' and 'field' in the 1980s and 'weights' on 1998. Finally, note the interesting relationship between the word 'kernel' and both 'Scholkopf' and 'Smola'.

6 Discussion and Future Work

In this paper we addressed the problem of learning time-varying binary pairwise-MRF given a set of time-stamped samples from the evolving network along its evolution path. We casted this high-dimensional temporal graphical model selection as a temporally smoothed L_1-regularized logistic regression problem (LTR), and we showed that the resulting problem can be handled by existing convex optimization techniques for up to few hundred nodes. We considered various options for enforcing smoothed regularization out of which the $L_1 + L_1$ penalty seems to be reasonably the best alternative for sparsity recovery. We then used our approach to analyze the NIPS12 dynamic academic social network among authors and keywords and presented interesting observations that showed the promise of our approach. There are several directions for future work. First, while standard off-the-shelf interior point methods were used successfully in this work, developing a customized version for the $L_1 + L_1$ case along the method proposed in [3] is an important future work that would enable "warm" restart in solving our optimization problem for various regularization parameters. On the theoretical side, while the authors in [9] showed that the $L_1 + L_1$ penalty is asymptotically consistent for the small sample case, (i.e. fixing graph size P while increasing the number of samples), it is still an open problem to show at which rate the sample size should grow relative to the graph size P and evolution rate to achieve neighborhood consistency. This result has been established though for the Gaussian case in [17] recently and we would like to establish it for the discrete case as well. Moreover, while our approach focused on smoothly evolving graphs, as opposed to other predictive models as in [1], the model in [1] can be initialized by our solution to expedite it convergence rate.

References

[1] F. Guo, S. Hanneke, W. Fu and E. P. Xing, Recovering Temporally Rewiring Networks: A model-based approach, In ICML 2007

[2] M. Wainwright, P. Ravikumar, and J. Lafferty, High dimensional graphical model selection using l1-regularized logistic regression. In Neural Information Processing Systems, 2006.

[3] K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. Journal of Machine learning research, 2007.

[4] S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In NIPS, 2006.

[5] E. Airoldi, D. Blei, E. Xing, S. Fienberg. A latent mixed membership model for relational data. 3rd international workshop on Link discovery 2005.

[6] P. Sarkar and A. Moore. Dynamic social network analysis using latent space models. KDD 2006.

[7] A. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In ICML 2004.

[8] R. Tibshirani, M. Saunders, S. Rosset, and J. Zhu. Sparsity and smoothness via the fused Lasso. Journal of the Royal Statistical Society Series B, 67(1):91108, 2005.

[9] R. Tibshirani. Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society, Series B, 58(1):267288, 1996.

[10] Y. Kim, J. Kim, and Y. Kim. Blockwise sparse regression. Statistica Sinica, 16:375390, 2006.

[11] M. Yuan and L. Lin. Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, Series B, 68(1):4967, 2006.

[12] J. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. IEEE Transactions on Information Theory, 52(3):10301051, 2006.

[13] S. Hanneke and E.P Xing, Discrete Temporal Models of Social Networks, In proceedings of the Workshop on Statistical Network Analysis, the 23rd International Conference on Machine Learning (ICML-SNA 2006).

[14] O. Banerjee, L. El Ghaoui, A dAspremont. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. JMLR 2007.

[15] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming (web page and software). http://stanford.edu/ boyd/cvx, May 2008.

[16] S. Zhou, J. Lafferty, and L. Wasserman. Time Varying Undirected Graphs. In COLT 2008, to appear.

[17] T. Snijders. Markov Chain Monte Carlo estimation of exponential random graph models. Journal of Social Structure 3(2)2002.

[18] S. Wasserman, G. Robins. An Introduction to Random Graphs, Dependence Graphs, and p*. Models and Methods in Social Network Analysis. 2005.

[19] C. Li and H. Li. Network-constrained regularization and variable selection for analysis of genomic data. In Bioinformatic, 2008.

[20] A. Ahmed and E.P. Xing. A Constraint Optimization Framework for Efficient Inference in htERGM. In Workshop on Statistical Models of Networks. (NIPS WS 2007).

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Carnegie Mellon