A Deep Learning Approach to Link Prediction in Dynamic Networks

Xiaoyi Li, Nan Du, Hui Li, Kang Li, Jing Gao and Aidong Zhang
Computer Science and Engineering Department, State University of New York at Buffalo
Buffalo, 14260, U.S.A.
{xiaoyili, nandu, hli24, kli22, jing, azhang}@buffalo.edu

Abstract

Time varying problems usually have complex underlying structures represented as dynamic networks where entities and relationships appear and disappear over time. The problem of efficiently performing dynamic link inference is extremely challenging due to the dynamic nature in massive evolving networks especially when there exist sparse connectivities and nonlinear transitional patterns. In this paper, we propose a novel deep learning framework, i.e., Conditional Temporal Restricted Boltzmann Machine (ctRBM), which predicts links based on individual transition variance as well as influence introduced by local neighbors. The proposed model is robust to noise and have the exponential capability to capture nonlinear variance. We tackle the computational challenges by developing an efficient algorithm for learning and inference of the proposed model. To improve the efficiency of the approach, we give a faster approximated implementation based on a proposed Neighbor Influence Clustering algorithm. Extensive experiments on simulated as well as real-world dynamic networks show that the proposed method outperforms existing algorithms in link inference on dynamic networks.

1 Introduction

Dynamic networks can be used to describe groups whose dynamics change over time. Link prediction on this kind of networks tries to predict future information based on historical data. It is precisely this type of information that would be most valuable in applications such as national security, online recommendations, and organizational studies. Some other application examples of link prediction can be found in biological networks. For example, predicting interactions between molecules (e.g. proteins or genes) at a specific timestamp can help us better understand the temporal interaction between molecules. More importantly, it can provide hints to discover temporal core mechanisms, which could be used as marker to indicate the stage of a specific disease such as cancer.

However, three major challenges hinder the development of dynamic network analysis systems. First, large dynamic networks may be complicated by the high dimensionality of responses, large numbers of observations and complexity of the choices to be made among explanatory variables. Second, sparse dynamic networks are sensitive to noise. Precisely, noise-to-signal ratio can be easily changed on sparse networks. Third, nonlinear transformations over time are commonly seen in dynamic networks with seasonal fluctuations, but the cost for catching these non-linearities is expensive.

Over the past years, many efforts have been devoted to develop systems or tools to predict future link states based on historical data. However, state-of-the-art link inference methods suffer from the following two weaknesses.

First, most methods rely on heuristics such as counting common neighbors, etc. While these often work well in practice, their theoretical properties have not been thoroughly analyzed. Most of the heuristic methods (e.g. [1], [12], [20]) ignore dynamics and evolutionary patterns of networks but rather predict links from one static snapshot of the graph. However, graph datasets often carry important evolutional information such as the creation and deletion of nodes and edges, so dynamic network data should be viewed as a continuous time process [21].

Second, probability based models usually suffer from model capacity and computational problems. Linear probability based models like Hidden Markov Models (HMM)[3] cannot efficiently model dynamic data with high variance due to the simple and discrete states defined in the model. Linear dynamic systems (e.g. [6]) have more powerful hidden space but they cannot model the complex nonlinear dynamics created by nonlinear properties in data. To tackle the non-linearity challenge, many other probability based models are proposed. However, piecewise models (e.g. [14]) is usually unable to conduct exact inference and approximations are costly and difficult to evaluate. Gaussian Process based dynamic models (e.g. [11]) suffer from their computational expenses (cubic in the number of training examples for learning and quadratic for prediction or generation). Kernel Regression based models (e.g. [15], [16]) find non-linear relationship between random variables, but each prediction is still based on finding nearest neighbors. Locality sensitive hashing (LSH) [9] is commonly used to overcome the searching overhead,

but the information loss caused by the selection of nearest neighbors can not be omitted.

To overcome these difficulties, we propose a generative model, i.e. Conditional Temporal Restricted Boltzmann Machine (ctRBM), which inherits the advantages of Restricted Boltzmann Machine family. It has distributed hidden states which means it has an exponentially large state space to manage the complex nonlinear variations. It has conditional independent structure that makes it easy to plug in external features. The model also has a simple way to learn deeper layers of hidden variables which is guaranteed to improve the overall model. Even though maximum likelihood learning is intractable, we can find an efficient learning algorithm with linear running time that achieves good approximation. Apart from the computational benefits, the proposed model can very well fit the two well known assumptions in dynamic network realm. First, each node has a unique transitional pattern. Second, node's behavior is influenced by its local neighbors. We address these assumptions for each node in the deep learning structure by modeling two types of directed connections to the hidden variables: *temporal connections* from a node's historical observations, and *neighbor connections* from the expectation of its local neighbors' predictions. To alleviate the computation costs introduced by adding neighboring effect, we propose a neighbor influence clustering algorithm to bring down the prediction time.

Our contributions can be summarized as follows:

- We propose a generative model in exponential family for link prediction in dynamic networks. The proposed model integrates neighbor influence as adaptive bias into the model energy function, and it is able to approach the real transitional distribution in an exponentially large space with efficient updating algorithm.
- We show that the proposed method can efficiently identify ill-behaved individuals by analyzing time-varying reconstruction error patterns.
- We propose a Neighbor Influence Clustering algorithm which further reduces the computation cost in the prediction phase to $O(n)$ without reducing model capacity.
- We conduct extensive empirical studies and show that the proposed model is able to capture seasonal fluctuations in dynamic networks and thus outperform existing methods.

The rest of the paper is organized as follows: In the next section, we present the formal definition of the problem and describe the challenges. In Sections 3 the proposed method is presented. Extensive experimental results are shown in Section 4. We finally conclude the paper in Section 5.

Table 1: Datasets Properties

Network	Avg.	Max.	Total.
synA	32.7	366	10,000
synB	32.6	389	10,000
Robot	3.9	1611	2,483,776
Control	126.7	1888	13,483,584
Exposure	103.2	1742	13,483,584

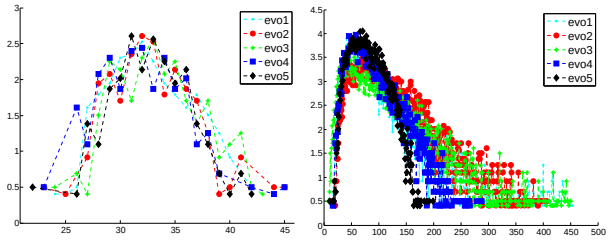

Figure 1: Examples of *DegreeChange* of *synB* (**left**), and *Control* (**right**). Y axis is the number of nodes in log scale (averaged between adjacent snapshots), X axis is the number of degree. A data point in this graph means the number of node changes (log scale) between adjacent snapshots in a particular degree slot.

2 Problem Definition and Challenges

In this section, we present problem definitions and illustrate the challenges in real dynamic networks.

2.1 Problem Definition Given a series of snapshots $\{G_1, G_2, ..., G_t\}$ of an evolving network $G_t = (\mathbb{V}, E_t)$, we seek to predict the most likely link state in the next time step, G_{t+1}. We assume that nodes $\mathbb{V}$ remain the same across all time steps but links E_t change for each time t.

2.2 Challenges In this paper, we conduct experiments on five dynamic networks with different level of difficulties in which the first two are synthetic networks and the remaining are real world networks. While details about the datasets and experiments can be found in Section 4.1, we first illustrate the challenges in dynamic networks that motivate our studies. The degree properties of these datasets are summarized in Table 1 where the second and third column exhibit the average and maximum degree of each network averaged over all the time steps, and the last column shows the total number of links in each network.

For most of the dynamic networks, sparseness measured by Node Degree varies from small to large. Different level of sparseness brings a big challenge to the

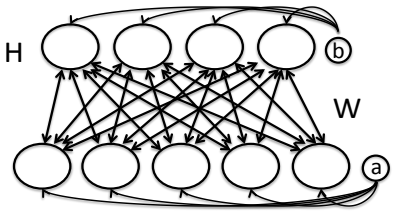

Figure 2: A Restricted Boltzmann Machine

model robustness. This difficulty also occurs in our dataset. i.e. Some node in *Robot* network shown in Table 1 has much sparser connections than others.

Nonlinear transformation is another challenge especially for biological dynamic networks. To visualize complicated transitional informations, we define *DegreeChange* between two time steps G_i and G_{i+k} as the absolute distance between the node degree distribution $Dist(G_i)$ and $Dist(G_{i+k})$, where $D = Dist(G)$ computes the degree distribution of a single snapshot G so that $D(i)$ is the number of nodes of degree i. An example transition shown in Figure 1 describes the pairwise difference ($k = 1$) between the first 6 snapshots (denoted as evo_X) in $synB$ and $Control$. As we can observe, transitional patterns cannot be easily found especially in the real data sets. It suggests that the proposed model should have the capabilities of modeling complex and nonlinear transition patterns.

3 Methodology

We have emphasized that models with distributed hidden state are necessary for efficiently modeling complex times series. However, this hidden states on directed models make the inference infeasible [2]. Therefore, if we use Restricted Boltzmann Machine (RBM) [8] based models, the posterior over latent variables factorizes completely which make the inference feasible. In this section, we first review the RBM and propose a new model – Conditional Temporal Restricted Boltzmann Machine (ctRBM) which is able to absorb temporal variations and neighbor opinions during the training phase, and make prediction conditioned on the current time-window and the expectation of local neighbors' prediction. The model maintains the most important computational advantages which make the training efficient using Contrastive Divergence algorithm [7].

The proposed method described in the following sections will train a ctRBM denoted as M_p for each node p, and finally get a collection of ctRBMs for all nodes denoted as $\mathbb{M}$. Therefore the final prediction, G_{t+1}, is obtained by collecting all the prediction results from each M_p. Although $\mathbb{M}$ makes a high space complexity, it can be easily deployed on large distributed platform. We will give further analysis in Section 4.

3.1 Restricted Boltzmann Machine Restricted Boltzmann Machine (RBM) is a special case of Markov Random Field which has two layers of variables: V and H forming a fully connected bipartite graph with undirected edges as shown in Figure 2. RBM defines a distribution over $(V, H) \in \{0,1\}^{N_V} \times \{0,1\}^{N_H}$, where V is the visible variables with dimension N_V and H the hidden variables with dimension N_H. This joint distribution is defined as:

$$P(V,H) = \exp(V'WH + a'V + b'H) \;/\; Z,$$
$$\text{where } Z = \sum_{V,H} \exp(V'WH + a'V + b'H). \tag{3.1}$$

In this equation, $W \in \mathbb{R}^{N_V \times N_H}$ is the weight between V and H, and the biases for V and H are a and b accordingly. Since there is no connection between nodes in each layer, the conditional distributions $P(H \mid V)$ and $P(\widetilde{V} \mid H)$ are fully factorial and given by

$$P(H_j = 1 \mid V) = \sigma(b_j + W'_{:,j}V),$$
$$P(\widetilde{V}_i = 1 \mid H) = \sigma(a_i + W_{i,:}H), \tag{3.2}$$

where σ is the logistic function defined as $\sigma(z) = (1 + exp(-z))^{-1}$, and i and j are the row and column index. $\widetilde{V}$ is the reconstructed data, representing the model's estimation. The goal of learning is to minimize the distance between $\widetilde{V}$ and V.

As with other models based on RBMs, the existence of the partition function Z makes maximum likelihood learning intractable. Nonetheless, it is easy to compute a good approximated gradient of an alternative objective function called Contrastive Divergence[7]. This approximation leads to a set of simple gradient update rules. The updates for all W, a, and b parameters take the form:

$$\Delta W_{i,j} \propto \langle V_i H_j \rangle_0 - \langle V_i H_j \rangle_K,$$
$$\Delta a_i \propto \langle V_i \rangle_0 - \langle V_i \rangle_K,$$
$$\Delta b_j \propto \langle H_j \rangle_0 - \langle H_j \rangle_K, \tag{3.3}$$

where $\langle \cdot \rangle_0$ defined as $\sum \boldsymbol{E}_{H|V}[\frac{\partial \boldsymbol{E}(V,H)}{\partial \theta}]$ is an expectation with respect to the data distribution, and $\langle \cdot \rangle_K$ defined as $\sum \boldsymbol{E}_{\widetilde{V},H}[\frac{\partial \boldsymbol{E}(V,H)}{\partial \theta}]$ is an expectation with respect to the joint distribution obtained from starting with a training vector clamped to the observations and performing K steps of alternating Gibbs sampling (i.e. CD-K). Typically K is set to 1, but recent results show that gradually increasing K during learning can significantly improve performance at an additional computational cost that is roughly linear in K [4].

3.2 Conditional and Temporal Property Nodes' transitional patterns and local neighbors' influence can be regarded as temporal and conditional property respectively. However, traditional RBM models can only

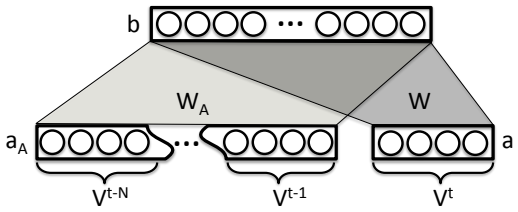

Figure 3: A Restricted Boltzmann Machine with temporal information, where N is the window size.

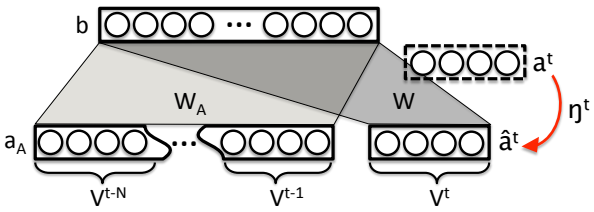

Figure 4: The summarized neighbor influence η^t is integrated into the energy function as adaptive bias.

model static frames of data. So for each node in the dynamic network, we propose to add two types of directed connections: **temporal connections** from the past N configurations (time steps) of the visible units to the current hidden configuration, and **neighbor connections** from the expectation of the local neighbors' predictions with respect to the the linkage history of the current node.

This reformation makes a new powerful generative model, in which weights for temporal connections can model local nonlinear temporal structure very well, leaving the neighbor connection weights to model global, macro-level structure.

3.2.1 Temporal Connections As shown in figure 3, we can model temporal dependencies by fixing the visible variables in the previous N time slice(s). There exist similar RBM based models (e.g. [18], [19]) which consider history states. However, they are targeting on object features instead of linkage status which made their assumptions and updating procedures very different from the proposed model.

In the temporal connections, N is a tunable parameter indicating how many time steps we need to look back. In modeling high time resolution data, we can set N to be related to the frame rate such that the proposed model will be able to deal with evolutionary data streams. To simplify the presentation, we will assume the data at $t-N$, ..., $t-1$ is concatenated into a vector $V^{<t}$ of dimension $N \cdot N_V$. Hence the weights for temporal connections are summarized by an $N \cdot N_V \times N_H$ weight matrix called W_A. By using this substitution, we avoid explicitly summing over past frames. Since the proposed model absorbs the temporary informations, the conditional probability will be changed to:

$$P(H^t \mid V^t, V^{<t}; \theta) = \alpha \cdot \sigma(b + W_A' V^{<t}) + (1-\alpha) \cdot \sigma(b + W' V^t) \tag{3.4}$$

$$P(\widetilde{V}^t \mid H^t) = \sigma(a + W H^t), \tag{3.5}$$

where W_A is the parameter for modeling the temporal variations. α is a hyper parameter balancing the model behavior between conservative and progressive, we set it to be 0.5 in this paper.

3.2.2 Neighbor Connections It is commonly known that a person's behavior will be affected by his/her friends circle. In our problem setting, neighbors' historical linkage behavior will indirectly affect the future linkage status of node p. This phenomenon can be simplified by the idea of Mean Field Theory that the effect of all the other individuals on any given individual is approximated by a single averaged effect. Similarly, we can summarize local neighbors' opinions into a unified one.

To formulate this idea, we define Neighbor Influence as the expectation of its local neighbors' predictions. If the total number of nodes in $\mathbb{V}$ is n, then the Neighbor Influence can be defined as:

$$\eta_p^t = \frac{1}{U_p^t} \sum_{q=1}^{n} l(x_p^t, x_q^t) \times P(\widetilde{V}_q^t \mid H_q^t) \cdot P(H_q^t \mid V_p^t, V_q^{<t}; \theta_q), \tag{3.6}$$

where $U_p^t = \sum_{q=1}^{n} l(x_p^t, x_q^t)$, and the indicator function l is 1 if node q is linking to node p at time t, and 0 otherwise. θ_q is the parameter of model M_q.

Neighbors make their predictions for node p base on their own experience (historical data). As we showed in Eq.(3.6), averaged opinion η_p^t of node p comes from its neighbors' opinions based on their own judgements (θ_q and $V_q^{<t}$). Since models are already trained by previous $t-1$ snapshots, $P(\widetilde{V}_q^t \mid H_q^t) \cdot P(H_q^t \mid V_p^t, V_q^{<t}; \theta_q)$ can be easily computed by Eq.(3.4)&(3.5) but substituting V^t in Eq.(3.4) by V_p^t.

3.3 Training and Inference on ctRBM Thanks to the complete factorable property over the latent variables, training and inference in the ctRBM is no more difficult than in the standard RBM. Specifically, the states of the hidden units are determined by both the input they receive from the individual observation V^t and $V^{<t}$, and the input η^t they receive from neighbors, shown in Figure 4. Given V^t and $V^{<t}$, the hidden units at time t are conditionally independent, as in Eq.(3.4). The effect of the Neighbor Influence can be viewed as adaptive bias:

$$\hat{a}^t = \beta \cdot a + (1-\beta) \cdot \eta^t$$

which includes the static bias, a for the current observation, and the contribution from the neighbors. β is a hyper parameter controlling individual to comply with themselves or their friends, we set it to be 0.5 in this work. This equation modifies the bias of visible units: a in Eq.(3.5) is replaced with $\hat{a}^t$ to obtain:

$$P(\widetilde{V}^t \mid H^t) = \sigma(\hat{a}^t + WH^t). \tag{3.7}$$

We can still use Contrastive Divergence for training the ctRBM. The updates for the weights have the same form as Eq.(3.3) but have a different effect because the states of the hidden units are now influenced by $V^{<t}$ and η^t. The gradients are now summed over all time steps and updated by the following rules:

$$\begin{aligned}
\Delta W_A &\propto \sum_t (\langle V^{<t} H^t \rangle_0 - \langle V^{<t} H^t \rangle_K), \\
\Delta a_A &\propto \sum_t (\langle V^{<t} \rangle_0 - \langle V^{<t} \rangle_K), \\
\Delta W &\propto \sum_t (\langle V^t H^t \rangle_0 - \langle V^t H^t \rangle_K), \\
\Delta b &\propto \sum_t (\langle H^t \rangle_0 - \langle H^t \rangle_K), \\
\Delta \hat{a} &\propto \sum_t (\frac{\beta}{1-\beta} \cdot (\langle V^{<t} \rangle_0 - \langle V^{<t} \rangle_K) + \eta^t).
\end{aligned} \tag{3.8}$$

While inferencing on a ctRBM, we do not need to proceed sequentially through the training data sequences. The future link status are only conditional on the past N time steps and the neighbor influence. As long as we fix the window size N, these small windows can be mixed and formed into mini-batches to speed up the learning process.

3.4 Inferencing Linkage by ctRBM After training, model parameters are fixed and predicting future link status follows the similar reason with inferencing. Specifically, we can shift the window one step towards future to obtain a fixed observation which contains the previous $N-1$ snapshots and the current snapshot. We fix this known observation as history, calculate neighbor influence based on a unified uncertainty (e.g. 0.5 for all units) and perform a forward inference to get the network status at $t+1$. The detailed procedure is shown in Algorithm 1.

As a property of generative models, a trained ctRBM can fill in missing data if there are dropouts occur in a sequence, which may save us lots of time from data preprocessing. The process is very similar to prediction. Formally, assume here is a missing snapshot G_d at time d. Given the known data ($\{G_1 \sim G_{d-1}\}$ and $\{G_{d+1} \sim G_t\}$), we can first initialize their corresponding visible units, then initialize the missing data to a unified uncertainty (0.5 for all units) and alternate between stochastically updating the hidden and visible units, with the known visible states held fixed.

Algorithm 1 Predicting G_{t+1} by ctRBM

Input: A trained $\mathbb{M}$ for all nodes, in which $M_p \in \mathbb{M}$ is parameterized by $\theta_p : \{W_{Ap}, W_p, a_{Ap}, a_p, b_p\}$.
A series of snapshots $\{G_{t-N+1}, ..., G_t\}$, where N is the window size.

Output: G_{t+1}

$p \Leftarrow 1$, $G_{t+1} \Leftarrow zeros(length(\mathbb{V}), length(\mathbb{V}))$
for $p < length(\mathbb{V}) + 1$ **do**
  $V_p^{<t+1} \Leftarrow \{G_{t-N+1}^p, ..., G_t^p\}$
  $V_p^{t+1} \Leftarrow ones(1, length(\mathbb{V})) \times 0.5$
  Get neighbor indices: $Idx \Leftarrow find(G_t^p == 1)$
  Get neighbor models list: $\mathbb{M}_{nbr} \Leftarrow \mathbb{M}(Idx)$
  Calculate η_p^{t+1} by Eq.(3.6) given $\mathbb{M}_{nbr}$
  $a_p^{t+1} \Leftarrow a_p + \eta_p^{t+1}$
  Calculate $\widetilde{V}_p^{t+1}$ by Eq.(3.4) & (3.7) substituting V^t and $V^{<t}$ by V_p^{t+1} and $V_p^{<t+1}$
  $G_p^{t+1} \Leftarrow \widetilde{V}_p^{t+1}$
end for

3.5 Detecting Ill-behaved Nodes by ctRBM One intrinsic ability of ctRBM is to detect ill-behaved nodes. Because nodes with random behavior usually provide less information. They randomly start and end relationship with other nodes which sabotage the network stableness. Besides, nodes which add or delete a massive number of connections in a very short period of time may indicate something significant, such as a shocking event happening in social network, or a serious deterioration of cancer. It would be meaningful to find these nodes and events in real-time. However, linear models usually fails this task because there exist nonlinear changing patterns of node degrees.

We can easily adapt the proposed model to enable this functionality. For example, we can record the euclidean distance between the reconstruction $\widetilde{V}^t$ (calculated by Eq.(3.7)) and V^t as t varies. Specifically, if we let the first sample to "burn-in" the proposed model and start monitoring the distances for the rest, then will have a vector of size $t - N$ where N is the window size. We define this vector as ϵ. So, for each node p, there is an ϵ_p recoding its "error pattern" along the time dimension. Ideally, the numbers recorded in ϵ_p should monotonically decrease to a very small residual because the proposed model fits the data better as time increases. However, the reconstruction may not be perfect, i.e., there are abnormal behavior which disturb the reconstruction error.

As we aware, the majority of nodes are normal and predictable. Therefore, the information for the majority of items should also be reliable and consistent. We

thus derive the **Consistency Score** r_p for each node as following:

$$r_p = \psi(\epsilon_p, median(\epsilon)), \tag{3.9}$$

where $median(\cdot)$ returns the error pattern of the majority. ψ can be any distance function, we choose to use *element-wised euclidean distance* in this work. As a result, r_p will be a vector of the same length with ϵ_p. Under this measurement, the higher the score, the further the node is away from predictable.

We can simply rank the Consistency Score based on $\sum r_p$ to find top problematic nodes. The distinction between noise and sudden events is also easy among these problematic nodes, because the reconstruction errors are consistently high across almost all the time steps for noise, while only a small number of time steps for sudden events. In consequence, we can distinguish them by setting a threshold, e.g., if more than 50% of the elements in ϵ_p are larger than the corresponding elements in $median(\epsilon)$, we can claim that node p is noise.

3.6 Reducing the Prediction Time Algorithm 1 indicates that for each node, prediction is made by conditioning on its past and all its neighbor opinions. The historical data can be absorbed by a fixed times of matrix multiplication which is in constant time, while the "consulting" dynamically changes over time because the neighbors of the node are changing. An interesting finding is that in most of the real world datasets, node degrees are usually less than the square root of the total number of links. e.g., the maximum node degree shown in Table 1. The computational complexity in this case is $O(n^{\frac{3}{2}})$. However, in the worst case scenario (e.g. when a node connects to all other nodes), this complexity increases to $O(n^2)$ which is not very efficient especially when we want to do real-time prediction.

We address this problem based on a simple hypothesis, that the impact and influence from neighbors can be clustered. There are two commonly known reasons sustaining this hypothesis. First, if two nodes share similar neighbors, their Neighbor Influences will be similar. Second, group behavior is much more stable and predictable than individual behavior. The hypothesis implies that similar Neighbor Influences can be represented by a center influence, and this centroid will not change heavily over time. This idea can be implemented in a distributed fashion by the following procedure:

The tunable parameter k controls the updating interval, a rule of thumb is to set $k = log(frame\ rate)$ for real-time applications.

We name this procedure Neighbor Influence Clustering (NIC) algorithm. In a word, this distributed approximation algorithm reduce the computation cost by bucketing the neighbor influences into separate cluster centroids to avoid querying neighbor opinions for each time step. Theoretically speaking, the worst case running time didn't change when k equals to the *frame rate*. However, since k in practical is much smaller than the frame rate and irrelevant to the number of nodes, the running time is very close to TRBM, which is $O(n)$.

Algorithm 2 Neighbor Influence Clustering

1: From the server, cluster η^t using any available clustering algorithm after a few time steps from the beginning (e.g. $t = 5$). Keep the resulting centroids and node assignments as global.
2: Send the corresponding centroid to node p where p belongs to, and replace $\eta_p^t \sim \eta_p^{t+k}$ by the centroid.
3: At time $t + k + 1$, each node sends request to its neighbors to compute η^{t+k+1}, and find its nearest neighbor with the global centroids, then upload new assignments to the server.
4: Server updates node assignments and goes back to step 2.

The initial position of these centroids will not very much affect the performance as long as they are separated. For the simplicity reason, we choose to use PCA and Kmeans to do clustering for the experiments. A practical benefit of NIC is that it is deployable on large distributed platform to handle millions of nodes.

4 Empirical Study

We first introduce our datasets and measurements, then show that our algorithm outperforms several baselines in a variety of situations, such as noisy and seasonality in link formation. These findings are confirmed on several evolving networks based on the evaluation metrics introduced in Section 4.2.

4.1 Datasets We use the following five datasets in the experiments.

Synthetic data A (*synA*) is generated based on the assumption that there are some common clusters within the network. For each snapshot, we generate 100 nodes, which are divided into five clusters of 20 nodes each, and these nodes' linkage behavior is sampled from predefined distributions. If nodes belong to the same cluster, their linkage behavior is drawn from the same distribution. It is worth noticing that each cluster's distribution may be different with each other. Moreover, a controllable part of nodes are randomly chosen to be the ones sampling from outlier distributions. All the nodes are then shuffled with noises are added. From the 2^{nd} to the 19^{th} snapshots, edges are added/deleted randomly

with a higher probability p_{in} for within-cluster edges and a lower probability probability p_{out} for between-cluster edges. It is easy to see that the way of generating the synthetic data simulates the actual situation.

Synthetic data B ($synB$) follows the same generation procedure with $synA$, but additional *seasonal* fluctuation property is added. Time moves over a repeating sequence of seasons. i.e., pick $\{G_1, G_2, ..., G_4\}$ from $synA$ and append reversely to form a complete fluctuation cycle. Such that each cycle contains 7 snapshots, denoted as $\{G_1, G_2, ..., G_4, G_3, ..., G_1\}$. We generate 3 cycles in this way (19 snapshots in total) with noise added such that their fluctuations are not identical.

Robot.Net dataset (*Robot*) was crawled daily from the website Robot.Net since 2008. This dataset contains the interactions among the users of the website. For the experiments, we choose the first sampled snapshot in each year during 2007 to 2012 and denote the obtained six snapshots as R1 to R6. In this website, each user is labeled by the others as Observer, Apprentice, Journeyer or Master according to his/her importance in the website. Based on these labels, we divide the users into four object role classes: Observer, Apprentice, Journeyer and Master.

Two biological datasets (*Control* and *Exposure*) are obtained from Stevenson et al.[17]. They collected the gene expression data from two groups of rats: the rats (eight replicates) exposed to cigarette smoke (i.e. exposure group) and the rats (eight replicates) exposed to room air only (i.e. the control group). Various intervals up to eight months are used to identify the molecular changes induced by cigarette smoke inhalation that may drive the biological and pathological consequences leading to disease, such as asthma and lung cancer. The dataset includes eleven snapshots (1, 3, 5, 14, 21, 28, 42, 56, 84, 112 and 182 days). In this study, we focus on 3672 genes whose p-values (via t-test) are smaller than 0.05.

4.2 Result Presentation Dynamic link prediction task is essentially a very difficult binary prediction problem due to the sparseness of the social network. Given n nodes, we have a space of $n^2 - n$ possible links, among them only a very small fraction of links actually exists.

In our data, the existing links only constitute less than 0.15% of all possible links. This means *accuracy* is not a meaningful measure. In our data sets, by predicting an empty network (no links exist), we can always achieve accuracy over 85%. In order to clearly distinguish the performance of different models, we use two measurements to present our results:

- Receiver Operating Characteristics (ROC) curve and Area Under the Curve (AUC).
- Sum of Absolute Differences (SumD)

Receiver Operating Characteristics (ROC) curve describes the fraction of true positive rate versus the fraction of false positive rate, at various threshold settings. As we move alone the x-axis, the larger percentage of links are selected, and the percentage of actually occurring links that are in the selected links monotonically increases. When all links are selected the percentage of the selected links will be equal to one. Therefore the ROC curve always starts at (0, 0) and ends at (1, 1). An ROC curve describing a random prediction will result in a back diagonal line. The standard measure to assess the quality of the ROC curve is the Area Under the Curve (AUC) measure, which is strictly bounded between 0 and 1. The larger the AUC is, the better the model performs.

When the AUC scores are close between two models, we cannot easily tell the pros and cons between two models especially when the network is large. So we propose to use *Sum of Absolute Differences* (SumD) defined as $SumD = \sum_i^{\mathbb{N}} \mid \widetilde{G}_i - G_i \mid$, in which $\mathbb{N}$ denotes the total number of elements in G; $\widetilde{G}$ denotes the model prediction and G is the ground truth we have. It is a strict measurement which ensures the distinction between different models.

4.3 Baseline Methods: We compare our approach with the following baseline methods, which cover a wide variety of ways to recommend edges.

- Common Neighbours (CN) [12]: Two nodes are more likely to form a link if they have many common neighbors.
- Adamic-Adar (AA) [1]: This method refines the simple counting of common neighbors by assigning the less-connected neighbors higher weights.
- Katz [10]: This method gives decreasing weight based on number of walks starting from a given node.
- Naive Bayes (NB) [5]: This method assume independent relationship between any two nodes and learn the probability of the current network state given its history.
- Autoregressive with first order gaussian (AR1) [13]: This method learns a linear dependencies between output variable and its previous values.
- Temporal Restricted Boltzmann Machine (TRBM): A RBM based structure shown in Figure 3 which only considers the current and previous time steps.

In the above methods, **CN**, **AA**, **Katz** predict next time step $t + 1$ only based on the current time step

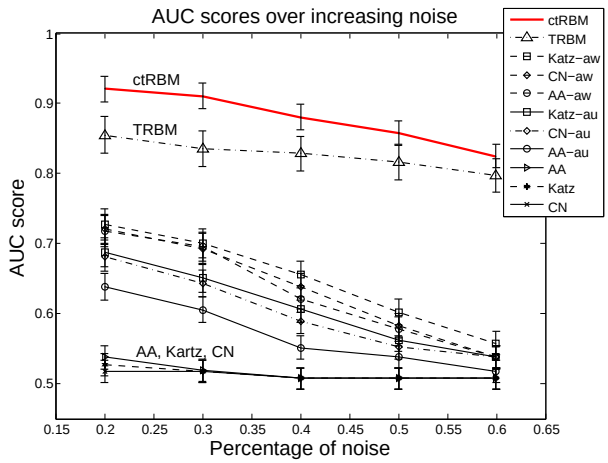

Figure 5: The performances on *SynA* with increasing percentage of noise added.

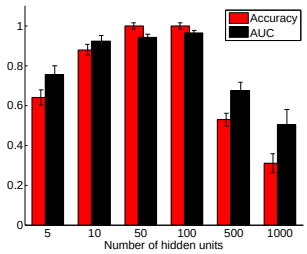

Figure 6: The performance on models with different expression capacity.

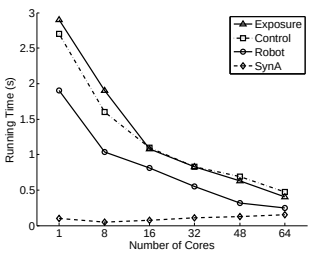

Figure 7: The speed up achieved by using more cores.

t. We denote their variations as **CN-au**, **AA-au** and **Katz-au**, where the notion **-au** means "all previous snapshots till time t are equally (unweighted) considered and used". We denote another variations of them as **CN-aw**, **AA-aw** and **Katz-aw** which use decaying weight for all previous snapshots when they are away from the current. **NB** and **AR1** are under the same situation as the models with "**au**".

4.4 Experiments We first test the noise tolerance of ctRBM against several baselines on distorted *SynA*. Specifically, we randomly flip the linkage of all nodes by a designated percentage. As shown in Figure 5, ctRBM consistently outperforms all the other baselines by a large margin. We observe that CN, AA and Katz behaves as bad as random predictor. This is because of their limitations of predicting $t+1$ only based on t, which is biased. TRBM also outperforms other heuristic baselines. However, the reason of performance gap exists between ctRBM and TRBM is because TRBM did not consider the global information.

Performance Test. Table 2 compares all the performances over the five datasets described in Section 4.1, in which **ctRBM-k** is the proposed model with Neighbor Influence clustered. We achieve the best among all baselines even if the network is very sparse (*Robot*), or with non-linear patterns (*SynB*). An interest finding is that AR1 performs very well on *Robot*. This is because the user interactions on Robot.Net evolve very slowly (changes are small and gradual), and the gaussian noise assumption fits the data. However, it has the worst performance on biological networks which have much more complicated variations. The missing items for *Control* and *Exposure* are simply because of the cubic time complexity.

Noisy Node Detection Test. To test our anomaly detection method, we manually pick some nodes in *SynA* and randomly flip their linkage behavior across all time. Since we know their indices, we can use accuracy to measure how many ill-behaved nodes are correctly captured. We find in Figure 6 that the proposed model can perfectly detect these nodes when the number of hidden units fit the data.

Parameter Sensitivity Test. The number of hidden units is closely related to the representational power of ctRBM. Figure 6 shows the performance of ctRBM under different parameter settings. When the number of hidden units is small, the model is lack of capacity to capture data complexity which results in lower performance on both detection accuracy and prediction. In contrast, the number is too large that we don't have sufficient samples to train the network which results in a even lower performance and stability. In fact, the parameter tuning is all about balancing model capacity with insufficient number of samples. In this case, we choose 100 to be the "sweet point".

Speedup Test. The proposed method (ctRBM with NIC) is implemented on a high-performance cluster containing a large number of computation units where each unit has CPU of 8 cores. We acquire 8 units and test our algorithm on 4 datasets. As shown in Figure 7, the running times keep decreasing when we involve more cores. However, the curve of *SynA* goes up after 8 because the inter-core communication time starts to dominate. Theoretically, we can always find a balance point which can maximize the performance while minimize the computation cost.

5 Conclusions

In this paper, we proposed a generative model – ctRBM for dynamic link prediction. The proposed model successfully captures the complex variations and nonlinear transitions such as seasonal fluctuations, and it can also perform direct inference to predict future linkage without nearest neighbor searches. The proposed Neighbor Influence Clustering (NIC) algorithm further re-

Table 2: Comparison with Baselines Using $SumD$ and AUC

	SynA	*SynB*	*Robot*	*Control*	*Exposure*	
baseline	*AA*	2424 (0.7125)	2296 (0.7206)	6158 (0.5016)	368101 (0.6342)	524511 (0.5711)
	AA-au	2702 (0.7127)	2368 (0.7125)	6157 (0.5016)	-	-
	AA-aw	2808 (0.7103)	2488 (0.7179)	6158 (0.5016)	-	-
	CN	3314 (0.5059)	3178 (0.5243)	6158 (0.5000)	643326 (0.5303)	902037 (0.5392)
	CN-au	2078 (0.7230)	2828 (0.5801)	6157 (0.5000)	-	-
	CN-aw	1954 (0.7024)	2848 (0.5766)	6158 (0.5000)	-	-
	Katz	2306 (0.7232)	2292 (0.7195)	6140 (0.5011)	369450 (0.6138)	528556 (0.5389)
	Katz-au	2168 (0.7291)	1874 (0.7531)	6142 (0.5010)	-	-
	Katz-aw	2194 (0.7318)	1930 (0.7474)	6140 (0.5011)	-	-
	NB	3109 (0.5704)	3314 (0.5831)	6209 (0.5000)	365405 (0.6545)	519117 (0.5409)
	AR1	2993 (0.6056)	1775 (0.8680)	19 (0.991)	787301 (0.5054)	940974 (0.5021)
	TRBM	384 (0.9322)	684 (0.9109)	0 (1.000)	76648 (0.7902)	85944 (0.7302)
proposed approach	*ctRBM*	**210 (0.9641)**	**411** (0.9427)	0 (1.000)	**61302 (0.8103)**	**69912 (0.7882)**
	ctRBM-k	259 (0.9574)	464 **(0.9441)**	0 (1.000)	68843 (0.8028)	78843 (0.7727)

duces the practical running time of prediction to $O(n)$. The measurement defined in NIC can also be applied on transitional patterns so that the proposed ctRBM model can be used to detect malfunction nodes easily. Experimental results on synthetic and real world datasets show that ctRBM outperforms existing link prediction algorithms on dynamic networks. The proposed ctRBM model provides a powerful analytic tool for understanding the transitional and evolutionary behavior of dynamic networks.

References

[1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social networks*, 2003.

[2] C. C. Aggarwal, Y. Xie, and S. Y. Philip. On dynamic link inference in heterogeneous networks. In *SDM*, 2012.

[3] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 1966.

[4] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *AISTATS*, 2005.

[5] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, 2006.

[6] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural computation*, 2000.

[7] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

[8] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[10] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953.

[11] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *NIPS*, 2004.

[12] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *JASIST*, 2007.

[13] T. C. Mills. *Time series techniques for economists*. Cambridge University Press, 1991.

[14] V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *NIPS*, 2000.

[15] P. Sarkar, D. Chakrabarti, and M. Jordan. Nonparametric link prediction in dynamic networks. *arXiv preprint arXiv:1206.6394*, 2012.

[16] U. Sharan and J. Neville. Temporal-relational classifiers for prediction in evolving domains. In *ICDM*, 2008.

[17] C. S. Stevenson, C. Docx, R. Webster, C. Battram, D. Hynx, J. Giddings, P. R. Cooper, P. Chakravarty, I. Rahman, J. A. Marwick, et al. Comprehensive gene expression profiling of rat lung reveals distinct acute and chronic responses to cigarette smoke inhalation. *AJP-LUNG*, 2007.

[18] I. Sutskever and G. E. Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *AISTATS*, 2007.

[19] G. W. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *ICML*, 2009.

[20] T. Tylenda, R. Angelova, and S. Bedathur. Towards time-aware link prediction in evolving social networks. In *SNAKDD*, 2009.

[21] D. Q. Vu, D. Hunter, P. Smyth, and A. U. Asuncion. Continuous-time regression models for longitudinal networks. In *NIPS*, 2011.