NetCycle: Collective Evolution Inference in Heterogeneous Information Networks

Yizhou Zhang[1,2] Yun Xiong[1,2] Xiangnan Kong[3] Yangyong Zhu[1,2]

[1]Shanghai Key Laboratory of Data Science, Shanghai, China
[2]School of Computer Science, Fudan University, Shanghai, China
[3]Worcester Polytechnic Institute, Worcester, MA, USA
{yizhouzhang14, yunx, yyzhu}@fudan.edu.cn xkong@wpi.edu

ABSTRACT

Collective inference has attracted considerable attention in the last decade, where the response variables within a group of instances are correlated and should be inferred collectively, instead of independently. Previous works on collective inference mainly focus on exploiting the *autocorrelation* among instances in a *static* network during the inference process. There are also approaches on time series prediction, which mainly exploit the autocorrelation within an instance at different time points during the inference process. However, in many real-world applications, the response variables of related instances can co-evolve over time and their evolutions are not following a static correlation across time, but are following an internal *life cycle*. In this paper, we study the problem of *collective evolution inference*, where the goal is to predict the values of the response variables for a group of related instances at the end of their life cycles. This problem is extremely important for various applications, *e.g.*, predicting fund-raising results in crowd-funding and predicting gene-expression levels in bioinformatics. This problem is also highly challenging because different instances in the network can co-evolve over time and they can be at different stages of their life cycles and thus have different evolving patterns. Moreover, the instances in collective evolution inference problems are usually connected through *heterogeneous information networks*, which involve complex relationships among the instances interconnected by multiple types of links. We propose an approach, called *NetCycle*, by incorporating information from both the correlation among related instances and their life cycles. We compared our approach with existing methods of collective inference and time series analysis on two real-world networks. The results demonstrate that our proposed approach can improve the inference performance by considering the autocorrelation through networks and the life cycles of the instances.

KDD '16, August 13-17, 2016, San Francisco, CA, USA
 ISBN 978-1-4503-4232-2/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939742

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining

Keywords

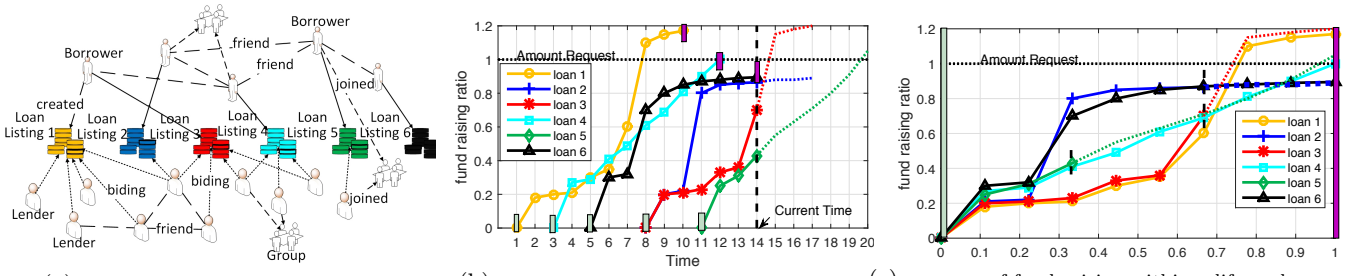
Collective Inference, Evolution Inference, Heterogeneous Information Networks, Graph Mining, Data Mining

1. INTRODUCTION

Conventional supervised learning usually assumes that the instances are independent and identically distributed (i.i.d.), where the response variables of different instances are inferred independently. In many relational data, however, the response variables of different instances can be related. For example, in financial networks, the loan listings borrowed by the same individual are more likely to share similar default risks than those borrowed by different people. An effective model for these relational data should be able to capture the dependencies among different instances during the inference process. Motivated by this challenge, collective inference problem [17, 15] has attracted much attention in recent years, where the response variables within a group of instances are correlated and should be inferred collectively.

In the literature, collective inference for both classification and regression problems, has been extensively studied [17, 15, 1, 10]. Previous works on collective inference mainly focus on exploiting the *autocorrelation* among instances in a *static* network during the inference process, as shown in Figure 2(a). However, in many real-world applications, the response variables of the instances can evolve over time, and the network can involve multiple kinds of relationships among the instances. For example, in *P2P Lending networks*, such as *Prosper.com* and *Kickstarter.com*, the financial activities of different loan listings can evolve over their entire fund-raising periods. These loan listings are interacting with different borrowers and lenders [14], as shown in Figure 1(a). In *PPI networks*, the expression-level of different genes can also change over time, which is regulated by their complex chemical reactions with different molecules [18]. Besides, *transportation networks*, such as road traffic networks and public transit networks, can also involve multiple evolving attributes, representing the traffic volumes of different road segments or subway stations.

There are many approaches on time series prediction, which mainly exploit the autocorrelation within an instance at different time points during the inference process [5, 12, 9].



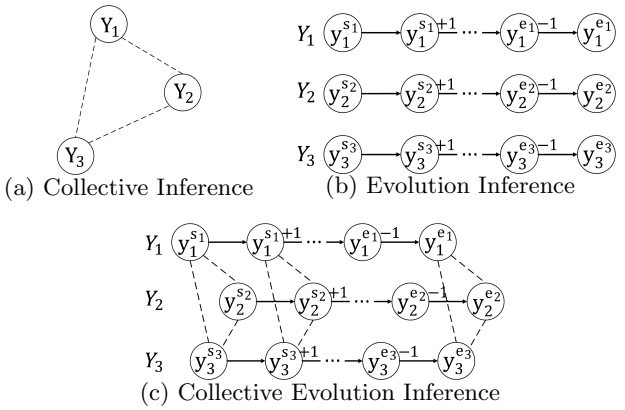
(a) Prosper P2P Lending Network (b) progress of fund-raising across time (c) progress of fund-raising within a life cycle

Figure 1: An example of collective evolution inference problem in P2P Lending networks. (a) an example of heterogeneous information network in P2P lending. (b) fund-raising progress of the loan listings across *time*. (c) fund-raising progress for the loan listings across fund-raising periods, *i.e.*, their *life cycles*.

However, in many real-world applications, the response variables of related instances can co-evolve over time and their evolutions are not following a static correlation across time, but are following an internal *life cycle*. For example, in the fund-raising networks, evolution of the fund-raising activities in a loan listing usually follows its fund-raising period (life cycle), which includes a predefined starting time and ending time. Activities of the loan listing may show different patterns at different stages of its life cycle, as shown in Figure 1(c). This problem can be called as *evolution inference*, where each instance can evolve within its life cycle, and the response variable is associated with a series of values. As shown in Figure 2(b), the goal of evolution inference is to estimate $y_i^{e_i}$.

In this paper, we study the problem of *collective evolution inference*, where the goal is to predict the values of the response variables for a group of related instances at the end of their life cycles. This problem is very important in various applications, *e.g.*, predicting fund-raising results in crowd-funding and predicting gene-expression levels in bioinformatics. Formally, *collective evolution inference* problem corresponds to predicting the response variables of a group of related instances at the end of their life cycles. This problem is highly challenging because different instances in the network can co-evolve over time and they can be at different stages of their life cycles and thus have different evolving patterns. We summarize the unique challenges of collective evolution inference as follows:

• **Collective Evolution with Life Cycles**: One major challenge of the collective evolution inference problem lies in the fact that the response variables values (response values for short) of the instances are evolving with a certain life cycle. Fortunately, the instances which have strong correlations are more likely to share similar evolution pattern during their life cycles. To better understand the collective evolution with life cycles, we give an example in Figures 1(b) and 1(c), which describes the evolution of the fund-raising ratio of the loan listings in P2P lending networks. As shown in Figure 1(b), the fund-raising ratio is evolving during fund-raising period. It is easy to think that the fund-raising ratio at time T depends on previous status at time $T-1$. However, the learning strategy of collective inference strictly follows the assumption that the response values of each instance will not change, which ignores the temporal information of the instances. Moreover, if we align the evolution of each loan listing by fund-raising period, as shown in Figure 1(c), the loan listings 1 and 3 which created by same borrower are more likely to obey similar fund-raising tendency during their life cycles. But collective inference problems usually focus on a "snapshot" of dynamic networks, while the snapshot captures the network at a certain time point. At a certain time point, the instances may be at different stages of their life cycles, which increased the difficulty of collective evolution inference.



(a) Collective Inference (b) Evolution Inference

(c) Collective Evolution Inference

Figure 2: Comparison of different inference problems. Y_i denotes the response variable of the i-th instance. y_i^k denotes the value of response variable Y_i at time point k. Here s_i/e_i is the starting/ending time point of the life cycle in the i-th instance. The instances directly linked by a dotted line are related.

• **Heterogeneous Dependencies**: Another challenge of collective evolution inference is the complex correlations among instances in HINs. It turns out strongly connected instances are more likely to show up with similar growth/evolving patterns in the same stages of their life cycles. The fundamental problem is how to effectively use the information provided by these multiple types of nodes and links, as well as the evolving tendency of each node. Collective inference approaches usually focus on homogeneous networks[1, 16] with one type of links and nodes. It is necessary to study how to exploit the complex dependencies among nodes' evolution in their life cycles in HINs, in order to predict the related instances more effectively.

In this paper, we first present collective evolution inference problem, which extends collective inference problem into dynamic networks, with the concept of life cycle. Then, we design a novel algorithm to solve the collective evolution inference problem, called NetCycle, which collectively infer the final values of a collection of nodes in a dynamic heterogeneous network by modeling the evolution tendency of the response variable. Finally, the experiments on real-world networks indicate the effectiveness of our algorithm. To our

Table 1: Important Notations.

Symbol	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	A heterogeneous network.
$\mathcal{V} = \{\mathcal{V}_1, \cdots, \mathcal{V}_K\}$	The set of nodes, involving K types of nodes. The target node type is $\mathcal{V}_1$.
$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i, s_i, e_i)\}_{i=1}^n$	The data set with n instances, $\mathcal{D} = \mathcal{V}_1$.
$\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$	The set of inherent attributes for all instances in $\mathcal{D}$.
$\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$	The set of response variables for all instances in $\mathcal{D}$.
s_i and e_i	The starting time and ending time of life cycle of i-th instance in $\mathcal{D}$.
$\mathbf{y}_i = \left(y_i^{s_i}, y_i^{s_i+1}, \cdots, y_i^{e_i}\right)$	A time series of values for the response variable Y_i of the i-th instance in $\mathcal{D}$.
T and t	T denotes current time, while t can indicates any time.
$\mathcal{L}$ and $\mathcal{U}$	The training set and testing set, where $\mathcal{L} \bigcup \mathcal{U} = \mathcal{D}$ and $\mathcal{L} \bigcap \mathcal{U} = \emptyset$.
$\mathcal{U} = \bigcup_{k=1}^M \mathcal{U}_k$	For $\forall i \in \mathcal{U}_k$, the evolution process of Y_i is only observed on the first k stages of its life cycle at time T, *i.e.*, $\mathbf{y}_i^k = \left(y_i^{s_i}, \cdots, y_i^{s_i+k-1}\right)$.
$\mathcal{N}_i$	The instances set which correlated with node v_{1i}. $\mathcal{N}_i \subseteq \mathcal{V}_1$.
$S = \{\mathcal{P}_1, \cdots, \mathcal{P}_{\lvert S\rvert}\}$	The set of meta paths type. Each $\mathcal{P}_l$ denotes a composite relationship between instances in $\mathcal{V}_1$.

best knowledge, this kind of problem has not been studied in the literature until recently.

2. PROBLEM DEFINITION

In this section, we first introduce some related concepts and notations, then define the problem.

2.1 Evolution Inference

Suppose we have a data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i, s_i, e_i)\}_{i=1}^n$ with n instances. Here $\mathbf{x}_i \in \mathbb{R}^d$ corresponds to the feature vector of the i-th instance in the d-dimensional space. $\mathbf{y}_i = \left(y_i^{s_i}, y_i^{s_i+1}, \cdots, y_i^{e_i}\right)$ corresponds to a time series of values for the response variable Y_i of the i-th instance. Y_i is evolving between time s_i (starting time) and e_i (ending time). $\forall t \in [s_i, e_i], y_i^t$ corresponds to the value of Y_i at time t. Here the response variable Y_i is only evolving during the period of time $[s_i, e_i]$, *i.e.*, the *life cycle* of the i-th instance. s_i is the starting time of the life cycle for Y_i, when the response variable Y_i starts to evolve. e_i is the ending time of the life cycle for Y_i, after which the response variable Y_i stops changing. Each instance in $\mathcal{D}$ can have a different life cycle, with different starting time and ending time. $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ represents the set of features for all instances in $\mathcal{D}$. $\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$ represents the set of target values for all instances in $\mathcal{D}$.

The indices of the instances in $\mathcal{D}$ are then divided into a training set $\mathcal{L} \subset \{1, \cdots, n\}$ and a test set $\mathcal{U} \subset \{1, \cdots, n\}$, where $\mathcal{L} \cup \mathcal{U} = \{1, \cdots, n\}$ and $\mathcal{L} \cap \mathcal{U} = \emptyset$. $\forall i \in \mathcal{L}$, the evolution of the response variable Y_i, *i.e.* $\mathbf{y}_i$, is fully observed. While in the test set $\mathcal{U}$, $\forall i \in \mathcal{U}$, the evolution of the response variable Y_i is only partially observed with different fractions. Suppose the life cycle of each instance has M stages, based upon the fraction of the evolution process being observed, the test set $\mathcal{U}$ can be divided into M disjoint groups, *i.e.*, $\mathcal{U} = \bigcup_{k=1}^M \mathcal{U}_k$. For $\forall i \in \mathcal{U}_k$, the evolution process of Y_i is only observed on the first k stages of its life cycle, *i.e.*, $\mathbf{y}_i^k = \left(y_i^{s_i}, \cdots, y_i^{s_i+k-1}\right)$. $\forall 1 \leq k \leq M$ and $\forall i \in \mathcal{U}_k$, the task of **evolution inference** on the i-th instance is to predict the value of Y_i at the end of its life cycle, *i.e.*, $y_i^{e_i}$, based upon $\mathbf{x}_i$ and $\mathbf{y}_i^k$. We use $\mathcal{Y}_\mathcal{U}^e = \{y_i^{e_i} | \forall i \in \mathcal{U}\}$ to denote all the target values for prediction. We then use $\mathcal{Y}_\mathcal{U}^o = \{\mathbf{y}_i^k \mid \forall 1 \leq k \leq M, i \in \mathcal{U}_k\}$ to denote all the observed evolution process in the test set and $\mathcal{Y}_\mathcal{L} = \{\mathbf{y}_i \mid \forall i \in \mathcal{L}\}$ to denote all the evolution process in the training set. The overall goal of *evolution inference* is to estimate the probability distribute $\Pr\left(\mathcal{Y}_\mathcal{U}^e | \mathcal{X}, \mathcal{Y}_\mathcal{L}, \mathcal{Y}_\mathcal{U}^o\right)$.

In this work, we simplify the problem by assuming the lengths of all life cycles are equal, *i.e.*, each instance has the same length in its life cycle. This assumption can be easily extended to unequal-length cases by normalization.

2.2 Collective Evolution Inference

To estimate $\Pr\left(\mathcal{Y}_\mathcal{U}^e | \mathcal{X}, \mathcal{Y}_\mathcal{L}, \mathcal{Y}_\mathcal{U}^o\right)$, conventional inference approaches usually require i.i.d. assumptions, and ignore dependency between different instances. The inference for each instance is performed independently:

$$\Pr\left(\mathcal{Y}_\mathcal{U}^e | \mathcal{X}, \mathcal{Y}_\mathcal{L}, \mathcal{Y}_\mathcal{U}^o\right) \propto \prod_{i \in \mathcal{U}} \Pr(y_i^{e_i} | \mathbf{x}_i)$$

Link-based Dependency

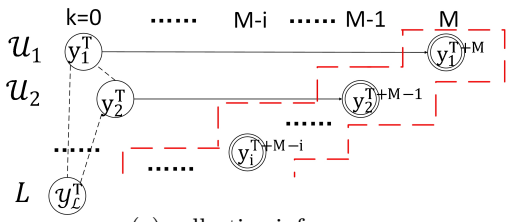
Collective inference approaches assume that the instances which directly linked in the network are related [1]. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ is the set of instances (nodes), $\mathcal{V} = \mathcal{D}$. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links between instances. We use T to represent the current time. $\forall v_i \in \mathcal{V}$, collective inference methods can model $\Pr(y_i^{e_i} | \mathcal{Y}_{\mathcal{N}_i}^T, \mathbf{x}_i)$. Here $\mathcal{Y}_{\mathcal{N}_i}^T$ contains all variables $Y_j (\forall v_j \in \mathcal{N}_i)$ at time T, *i.e.*, $\mathcal{Y}_{\mathcal{N}_i}^T = \{y_j^T | v_j \in \mathcal{N}_i\}$, and $\mathcal{N}_i$ denotes the instances set which correlated with v_i, $\mathcal{N}_i \subseteq \mathcal{V}$. Hence, by considering the correlation between instances, we will have:

$$\Pr\left(\mathcal{Y}_\mathcal{U}^e | \mathcal{X}, \mathcal{Y}_\mathcal{L}, \mathcal{Y}_\mathcal{U}^o\right) \propto \prod_{v_i \in \mathcal{U}} \Pr(y_i^{e_i} | \mathcal{Y}_{\mathcal{N}_i}^T, \mathbf{x}_i) \tag{1}$$

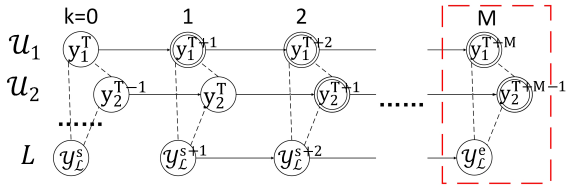
Collective Evolution with Life Cycles

Note that different instances may be at different stages of their life cycles at time T, so it is inappropriate to infer $y_i^{e_i}$ with $\mathcal{Y}_{\mathcal{N}_i}^T$, because the evolution tendency varies at different stages, which may mislead the inference result. Correlated instances share similar evolving pattern during their life cycles rather than during the absolute time. Suppose $v_i \in \mathcal{U}_k$, one better way is to replace $\mathcal{Y}_{\mathcal{N}_i}^T$ by using $\mathcal{Y}_{\mathcal{N}_i}^k$. Here $\mathcal{Y}_{\mathcal{N}_i}^k = \{y_j^{s_j+k} | v_j \in \mathcal{N}_i\}$, contains all response values $Y_j (\forall v_j \in \mathcal{N}_i)$ at k-th stage of their life cycles.

Moreover, with the assumption that the network is static, collective inference methods usually ignore the evolution information. In order to utilize this information, one way is to iteratively predict the target value $y_i^{s_i+k}$ based upon previous value $y_i^{s_i+k-1}$, until $s_i + k = e_i$, *e.g.*, autoregressive



(a) collective inference



(b) collective evolution inference

Figure 3: The basic idea of collective inference and collective evolution inference. Here k represents the k-th stage of life cycle. $\mathcal{U}_k$ is the instances set which stay in k-th stage of life cycle at current time T. The single circle means observed values of the response variable, and the double circle means predict values of the response variable.

model of order one (AR(1) for short). These methods can model $\Pr(y_i^{s_i+k}|y_i^{s_i+k-1}, \mathbf{x}_i)$, but the results usually have large errors when the prediction horizon is long, because it only utilizes previous evolution process. Note that the instances with strong correlations may obey similar evolution tendency during their life cycles, as shown in Figure 1(c). By considering both link-based dependencies and evolution information of instances, the idea of *collective evolution inference* is to model $\Pr(y_i^{s_i+k}|y_i^{s_i+k-1}, \mathcal{Y}_{\mathcal{N}_i}^k, \mathbf{x}_i)$, which utilizes the correlations between instances during their life cycles to reduce prediction errors. The task of collective evolution inference is to estimate:

$$\Pr\left(\mathcal{Y}_{\mathcal{U}}^e|\mathcal{X}, \mathcal{Y}_{\mathcal{L}}, \mathcal{Y}_{\mathcal{U}}^o\right) \propto \prod_{v_i \in \mathcal{U}} \Pr(y_i^{e_i}|y_i^{e_i-1}, \mathcal{Y}_{\mathcal{N}_i}^{e_i}, \mathbf{x}_i) \quad (2)$$

$$y_i^{s_i+k} = \begin{cases} \mathcal{F}\left(y_i^{s_i+k-1}, \mathcal{Y}_{\mathcal{N}_i}^k, \mathbf{x}_i\right) & T < s_i + k \le e_i \\ y_i^{s_i+k} & s_i + k \le T \end{cases}$$

Here $\mathcal{F}$ denotes the models which estimate response variable Y_i at time $s_i + k$, *e.g.*, AR(1).

Compared with collective inference, collective evolution inference avoids inferring $\mathcal{Y}_{\mathcal{U}}^e$ simultaneously, by aligning the life cycle of each instance into the same position. And it also makes use of the evolution process during the life cycles of instances. We use time T to indicate s_i and e_i ($\forall v_i \in \mathcal{V}$), as shown in Figure 3, we describe the difference between collective inference and collective evolution inference based upon Eq. 1, 2. Note that $\forall v_i \in \mathcal{U}_k$, $T = s_i + k - 1$, and $\forall v_i \in \mathcal{L}$, $T \ge e_i$, *i.e.*, $y_i^T = y_i^{e_i}$.

Heterogeneous Dependencies

In many real-world applications, the networks include multiple types of nodes and links, which are called *heterogeneous information networks* [10].

Definition 1. Heterogeneous information network is a special kind of information network, which can be represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ denotes the set of nodes, including K types of instances: $\mathcal{V}_1 = \{v_{11}, \cdots, v_{1n_1}\}, \cdots, \mathcal{V}_K = \{v_{K1}, \cdots, v_{Kn_K}\}$, where v_{ji} represents the ith instance of type j. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the links between the instances in $\mathcal{V}$, which involves multiple types of links.

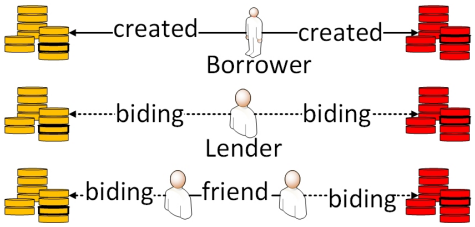


Figure 4: Examples of different types of Meta-path between two loan listings in P2P Lending Network.

For example, as shown in Figure 1(a), the P2P Lending network includes multiple types of nodes, *e.g.*, loan listing, borrower, lender, group, which are connected through multiple types of links, *e.g.*, created, bidding, friend, joined. In this paper, we focus on studying the collective evolution inference problem on one type of nodes, instead of on all of them in HINs. The reason is different type of nodes have different variables in HINs, so it's unreasonable to assume all types of nodes share the same set of response variables.

Without loss of generality, we define the first node type $\mathcal{V}_1$ as the target node type, *i.e.*, $\mathcal{V}_1 = \mathcal{D}$. By mining the linkage structure of HINs, multiple types of dependencies among instances can be extracted. In next section, we propose a method to solve the collective evolution inference in HINs. We summarized all notations in Table 1.

3. PROPOSED METHOD

In this section, we first introduce a concept named *meta path* [11] in section 3.1, which is often used to extract complex relationships among the instances in HINs. Then, we discuss how to solve the collective evolution inference problem in HINs based on meta path in section 3.2, and propose a simple and effective algorithm for meta path-based collective evolution inference in section 3.3. Finally, we discuss how to predict the node evolution process during the life cycle in section 3.4.

3.1 Meta Path

The *meta path* is defined as a sequence of relations in the network schema. The instances in HINs are inter-connected through multiple types of links. Each type of links from node type $\mathcal{V}_i$ to node type $\mathcal{V}_j$ corresponds to a binary relation R, where $R(v_{ip}, v_{jq})$ holds if the object v_{ip} and v_{jq} are linked in R. For example, in Figure 1(a), the link type "bidding" is a relation between *lender* nodes and *loan listing* nodes, where $R(v_{ip}, v_{jq})$ holds if the *lender* node v_{ip} has a link of type "bidding" to the *loan listing* node v_{jq} in the network. We can write the link type as "lender $\xrightarrow{bidding}$ loan listing".

A meta path $\mathcal{P}$ corresponds a type of path within the network, containing a certain sequence of link types. For instance, we give three types of meta path between *loan listings* in Figure 4, *e.g.*, a meta-path "loan listing $\xrightarrow{bidding^{-1}}$ lender $\xrightarrow{bidding}$ loan listing" denotes the composite relationship between *loan listing* nodes, where the semantic meaning of this meta-path is that the two *loan listing* nodes had been invested by same *lender* nodes. Here the link type "bidding^{-1}" represents the inverted relation of "bidding".

Input:
$\mathcal{G}$: a heterogeneous information network, M: number of stages in a life cycle, p_max: maximum meta-path length (default=4).
$\mathcal{X}$: attribute vectors for all instances, A: a base learner for local model, Max_{It}: maximum # of iteration (default=10)

Meta-path Construction:
- Construct the meta-path set $S = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|S|}\}$:
  Breadth search on schema graph of $\mathcal{G}$, starting from V_1 by adding short meta-path $\mathcal{P}_l$ that ends with V_1 into S first:
  1. If the length of meta-path $\mathcal{P}_l$ is greater than p_max, exit the BFS;
  2. If current meta-path $\mathcal{P}_l$ cannot be reconstructed by the paths in $\mathcal{S}$, then add P_l into S; Otherwise, prune the current path from BFS.

Segment Training:
- For each stage $k = 1, \cdots, M$, learn the segment model f_k:
  1. Construct an extended training set $\mathcal{D}^k = \{(\mathbf{x'}_i^k, y_i^{s_i+k})\}$ by converting each instance $\mathbf{x}_i$ to $\mathbf{x'}_i^k$ as follows:
     $\mathbf{x'}_i^k = \left(\mathbf{x}_i, y_i^{s_i+k-1}, \text{PathRelFeature}(v_{1i}, \mathcal{Y}_{\mathcal{L}}^k, S)\right)$. Note that $y_i^{s_i-1} = [\,]$
  2. Let $f_k = A(\mathcal{D}^k)$ be the segment model trained on $\mathcal{D}^k$.

Collective Evolution Inference:
- For each stage $k = 1, \cdots, M$:
  Bootstrap: Estimate the response values sets in k-th stage, for each $v_{1i} \in \bigcup_{j=1}^{k} \mathcal{U}_j$, suppose $v_{1i} \in \mathcal{U}_j$, produce an estimated values y_i^{T-j+k} (*i.e.*, $y_i^{s_i+k}$) as: $y_i^{T-j+k} = f_k\left((\mathbf{x}_i, y_i^{T-j+k-1}, \mathbf{0})\right)$. Note that $y_i^{T-j} = [\,]$.
  Iterative Inference: Repeat until convergence or #iteration $> Max_{It}$
  1. Construct the associate features $\mathbf{x}'_i$ for each testing instance $v_{1i} \in \bigcup_{j=1}^{k} \mathcal{U}_j$) as follows:
     $\mathbf{x'}_i^{T-j+k} = \left(\mathbf{x}_i, y_i^{T-j+k-1}, \text{PathRelFeature}(v_{1i}, \mathcal{Y}_{\mathcal{L}}^k \bigcup \{y_i^{T-j+k} | v_{1i} \in \bigcup_{j=1}^{k} \mathcal{U}_j\}, S)\right)$.
  2. Update y_i^{T-j+k} as $y_i^{T-j+k} = f_k(\mathbf{x'}_i^{T-j+k})$ for each testing instance $v_{1i} \in \bigcup_{j=1}^{k} \mathcal{U}_j$ (Suppose $v_{1i} \in \mathcal{U}_j$).

Output: $\mathcal{Y}_{\mathcal{U}}^e = \left(y_1^{e_1}, \cdots, y_{|\mathcal{U}|}^{e_{|\mathcal{U}|}}\right)$: the response values sets of testing instances ($v_{1i} \in \mathcal{U}$).

Figure 7: The NetCycle algorithm

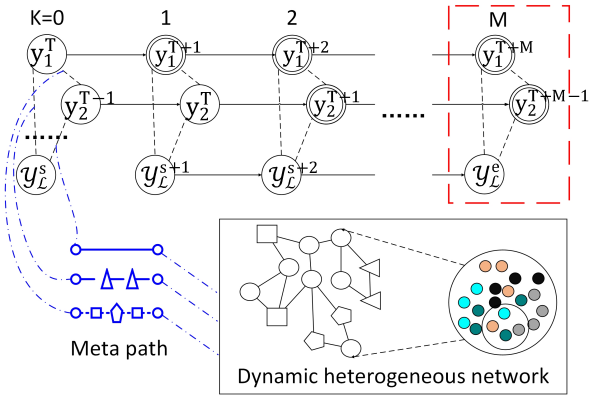


Figure 5: Meta path-based collective evolution inference in heterogeneous networks.

- $\mathcal{Y}_{\mathcal{N}_i}^k = \text{PathRelFeature}\left(v_{1i}, \mathcal{Y}^k, S = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|S|}\}\right)$

For each meta-path $\mathcal{P}_l \in S$:
1. Get related instances for node v_{1i} through meta path $\mathcal{P}_l$, *i.e.*, the related index set $\mathcal{C} = \mathcal{P}_l(i)$
2. $\mathcal{Y}_{\mathcal{P}_l(i)}^k = Aggregation\left(\{y_j | v_{1j} \in \mathcal{C}\}\right)$

Return $[\mathcal{Y}_{\mathcal{P}_1(i)}^k, \cdots, \mathcal{Y}_{\mathcal{P}_{|S|}(i)}^k]^{\mathrm{T}}$

Figure 6: The function of constructing relational features(PathRelFeature).

3.2 Meta path-based Collective Evolution Inference

In HINs, there are complex dependencies not only among instances directly linked through links, but also among instances indirectly linked through different meta paths. In order to solve the collective evolution inference problem more effectively, we explicitly consider different types of meta path-based dependencies among instances.

Meta path-based dependencies refer to the dependencies among instances that are inter-connected through a meta path, *e.g.*, the co-borrower relation between loan listings can be represented as the meta-path "loan listing $\xrightarrow{created^{-1}}$ borrower $\xrightarrow{created}$ loan listing". Most relationships studied in network data can naturally be captured by different meta paths. Given a set of meta paths $S = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|S|}\}$, $\mathcal{P}_l(i)$ denotes the nodes set where the nodes relate to node v_{1i} through the meta path $\mathcal{P}_l$, $\mathcal{P}_l(i) = \{v_{1j} | (v_{1i}, v_{1j}) \in \mathcal{P}_l\}$. The meta path-based dependencies can be used as follows:

$$\Pr\left(\mathcal{Y}_{\mathcal{U}}^e | \mathcal{X}, \mathcal{Y}_{\mathcal{L}}, \mathcal{Y}_{\mathcal{U}}^o\right) \propto \prod_{v_i \in \mathcal{U}} \Pr(y_i^{e_i} | y_i^{e_i-1}, \mathcal{Y}_{\mathcal{P}_1(i)}^{e_i}, \cdots, \mathcal{Y}_{\mathcal{P}_{|S|}(i)}^{e_i}, \mathbf{x}_i) \tag{3}$$

$$y_i^{s_i+k} = \begin{cases} \mathcal{F}\left(y_i^{s_i+k-1}, \mathcal{Y}_{\mathcal{P}_1(i)}^k, \cdots, \mathbf{x}_i\right) & T < s_i + k \le e_i \\ y_i^{s_i+k} & s_i + k \le T \end{cases}$$

Here $\mathcal{Y}_{\mathcal{P}_l(i)}^k = \{y_j^{s_j+k} | v_j \in P_l(i)\}$, denotes the response values set in k-th stage of life cycle, which the instances belong to $\mathcal{P}_l(i)$. The basic idea of meta path-based collective evolution inference as shown in Figure 5. An aggregation function (*e.g.*, mean, count, etc.) is often used to extract relational features from meta paths-based correlated variables set [11], as shown in Figure 6.

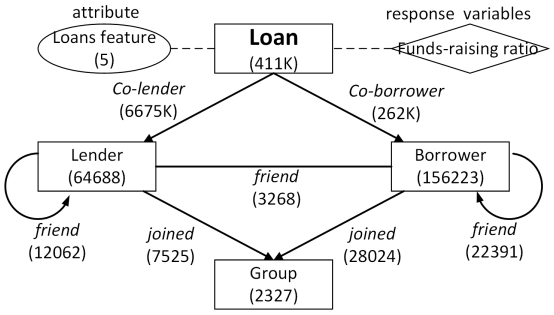
3.3 NetCycle Algorithm

In this section, we propose an algorithm, called NetCycle, to estimate Eq. 3. Inspired by the success of ICA framework [1] in collective inference, we designed a similar inference procedure for our NetCycle method as shown in Figure 7. The algorithm includes the following steps:

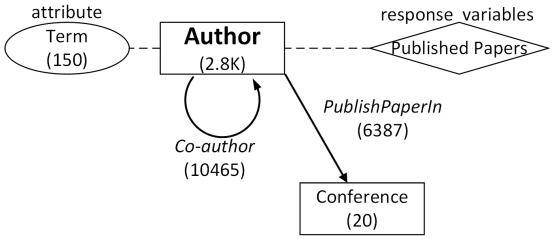
Meta-path Construction: Given a HIN, we first extract all non-redundant meta-paths for correlations of target instances type separately. A meta-path $\mathcal{P}_l$ in S is non-redundant if $\mathcal{P}_l$ cannot be reconstructed by combining any subset of the meta-paths in S. We only extract short meta-paths with a maximum path length p_max. It has been shown in [20] that long meta-paths are not quite useful in capturing the linkage structure of HINs.

Segmentation Training: We construct M extended training sets $\forall 1 \le k \le M, \mathcal{D}^k = \left\{(\mathbf{x'}_i^k, y_i^{s_i+k})\right\}$ by converting each instance $\mathbf{x}_i$ to $\mathbf{x'}_i^k$ using the functions in Figure 6, and combining with previous value $y_i^{s_i+k-1}$. Here $\mathcal{Y}^k = \{y_j^{s_j+k} | v_{1j} \in \mathcal{V}_1\}$, contains all response variables $Y_j (\forall v_{1j} \in \mathcal{V}_1)$ at k-th stage of their life cycles in Figure 6. We use base learner to train one segment model on each stage, by using the extended training sets.

Collective Evolution Inference: Overall, it is an ex-



(a) Prosper Datasets



(b) DBLP Dataset

Figure 8: Schema of datasets. The numbers of each type are tagged under the corresponding type.

tension of iterative inference algorithm [1] for the inference step. During the inference, we iteratively update the response variables set predictions of the testing instances in k-th stage of life cycle, for each $v_{1i} \in \bigcup_{j=1}^{k} \mathcal{U}_j$. We use the predicted value y_i^{T-j+k} as one of input attribute in predict the response values set in $k+1$ stage, until $k = M$. In the end, we will get $\mathcal{Y}_{\mathcal{U}}^{e}$ for the testing instances.

Suppose the complexity of base learner A is C, the time complexity of learning the NetCycle model is $\mathcal{O}(p_{max} \cdot \mathcal{V}_1^3 + M \cdot C)$. While the time complexity of constructing meta paths is $\mathcal{O}(p_{max} \cdot \mathcal{V}_1^3)$, which usually constructed through matrix multiplication.

3.4 Extension: Node Evolution Process Prediction

During the collective evolution inference process, for each instance v_{1i} in test set $\mathcal{U}$, suppose $v_{1i} \in \mathcal{U}_k$, the NetCycle algorithm can not only infer the target value $y_i^{e_i}$, but also the whole time series values of response variable Y_i, *i.e.*, $\left(y_i^{s_i}, \cdots, y_i^{s_i+k}, y_i^{s_i+k+1}, \cdots, y_i^{e_i}\right)$. It can help to discover the evolution tendency over the instances' life cycle. Further, the weight of parameters learned by segment models f_k, can help to reflect the importance of each feature during life cycle (including inherent attributes, previous value and topology features). Greater weight means greater impact on evolution, so we can use it to find the major factor of node evolution in different stages. For example, in P2P lending network, previous studies [2] validated that the fund raising process is mainly influenced by the investment rate in early stage, and mostly affected by social interactions (*e.g.*, herding effect) in latter stage over fund-raising period.

4. EXPERIMENTS

4.1 Data Collection

In order to validate the collective evolution inference performances, we apply our methodology to three real-world HINs (Summarized in Table 2).

•**Prosper Dataset:** Our first two datasets studied in this paper were extracted from Prosper P2P Lending platform[1]. *Prosper.com* provides abundant lending information on website, including loan information, bidding, borrower and investor information, *etc.* The network schema of Prosper datasets is shown in Figure 8(a). In this network, there are four types of nodes: loan listing, borrower, lender and group, connected by four types of relations/links: *created*, *friend*, *bidding* and *joined* relationship. We treat loan listings as our target instances, with their funds-raising ratio as the response variables. Then the life cycle of the instances is the loan listing's fund-raising period. Each loan listing has five inherent attributes: maximum rate borrower willing to accept *br*, debt-to-income ratio *dr*, total amount requested *ta*, whether or not borrower is a home owner *h*, and loan description length *dl* [2]. We extract two subsets of loan listings in the network from 2006∼2011, and remove all isolated points (which did not receive any fund). The first dataset is extracted from Prosper: 2006∼2007, and the second from Prosper: 2008∼2009, each have $30K$ loan listings, we add $100K$ unobserved loan listings which randomly sampled from 2006∼2011 for each dataset. The fund-raising period of each loan is divided into M stages, here $M = 10$ for all experiments in these two datasets.

Table 2: Summary of datasets.

Dataset	Node Type	Link Type	Feature	Instance
Prosper: 2006 ∼ 2007	4	4	5	30K+ 100K(Unobserved)
Prosper: 2008 ∼ 2009	4	4	5	30K+ 100K(Unobserved)
DBLP Dataset	3	2	150	2792

•**DBLP Dataset:** The third dataset, *i.e.*, DBLP four areas [8], is a bibliography information network extracted from DBLP[2], which involves three types of nodes: conference, paper and author, connected by two types of relations/links: conference-author links and co-author links. We treat authors as our target instances, with the number of published papers as the response variables. In this dataset, the life cycle is defined as the first five years since the author published his/her first paper. The inference task is to predict the numbers of papers published in the first five years of each author's academic life, which we call the authors' academic career problem, because most PhD degrees take five years, *i.e.*, $M = 5$. For each author, we extracted a bag-of-words representation of all the paper titles published by this author as inherent attributes, which include 150 words (terms) as node features. In this paper, we select the authors who published more than 2 papers, and the network schema of DBLP dataset is shown in Figure 8(b). For detailed description of the DBLP dataset, please refer to [8].

4.2 Algorithms for comparison

In order to demonstrate the effectiveness of our methodology, we compared with the following state-of-the-art algorithms (summarized in Table 3):

[1]https://www.prosper.com/

[2]http://dblp.uni-trier.de/db/

Table 3: Types of models, based on the kinds of features used.

Method	Self attr.	Neigh. labels	Time info	Publication
LIBLINEAR	✓			[4]
Herding Para[3]	✓		✓	[2]
GNetMine		✓		[8]
HCC	✓	✓		[11]
HCC(with kernel)[4]	✓	✓	✓	[11, 19]
NetCycle(w/o network)	✓		✓	This paper
NetCycle	✓	✓	✓	This paper

• LIBLINEAR [4]: This method is the base learner for following algorithms. It supposes that each instance is independent in the network.
• Herding Para [2]: This method is only applied to Prosper dataset. This method extracts the temporal features of bidding behavior to predict whether the loan will raise enough funds with logistic regression. However, it is a verifying experiment which can only be used after fund-raising period. We use a horizontal line to represent this method's performance in each stage.
• GNetMine [8]: This is a graph-based transductive classification approach based on information propagation model, which makes prediction on heterogeneous networks. It only uses static network structure, and assumes each node type shared by a group of similar labels.
• HCC [11]: This is a collective classification approach, which works on heterogeneous network by exploiting dependencies based on multiple meta paths in the network. The same as our method, it focuses on predicting one type of nodes in heterogeneous networks.
• HCC (with kernel) [19]: This framework can model both temporal and relational dependencies in dynamic networks, by giving different weight with *temporal-relational kernel* to links in the temporal dimension. But the paper focuses on link prediction instead of collective inference problem. In order to compare with the method, we combine the idea of modeling dynamic networks with kernels and HCC method, to solve the collective inference problem in dynamic networks.
• NetCycle (w/o network): This is a weak version of our method without network structure. It only uses the inherent attributes of nodes and the previous values of the nodes response variables, and does not consider the dependencies between related instances.
• NetCycle: This is our method proposed in Section 3.3, which tries to solve collective evolution inference in HINs.

For a fair comparison, all above algorithms use LIBLINEAR [4]. We use L2 regularized logistic regression as the base classifier and support vector regression as the base regression model. The weight of regularized item λ is set to be 0.001, and the maximum number of iterations Max_{iter} is set to be 10. The evaluation metric of classification is accuracy, and the evaluation metric of regression is root mean square error (RMSE). For all experiments, we use $k = 0/k = 1$ represents the start/end stage of nodes life cycles. 5-fold cross validations are used in all experiments.

[3]Prosper datasets only.
[4]Sharan et al. proposed an original way to model temporal network, however, they focus on the link prediction problem, we combined the idea with HCC as one of compared method.

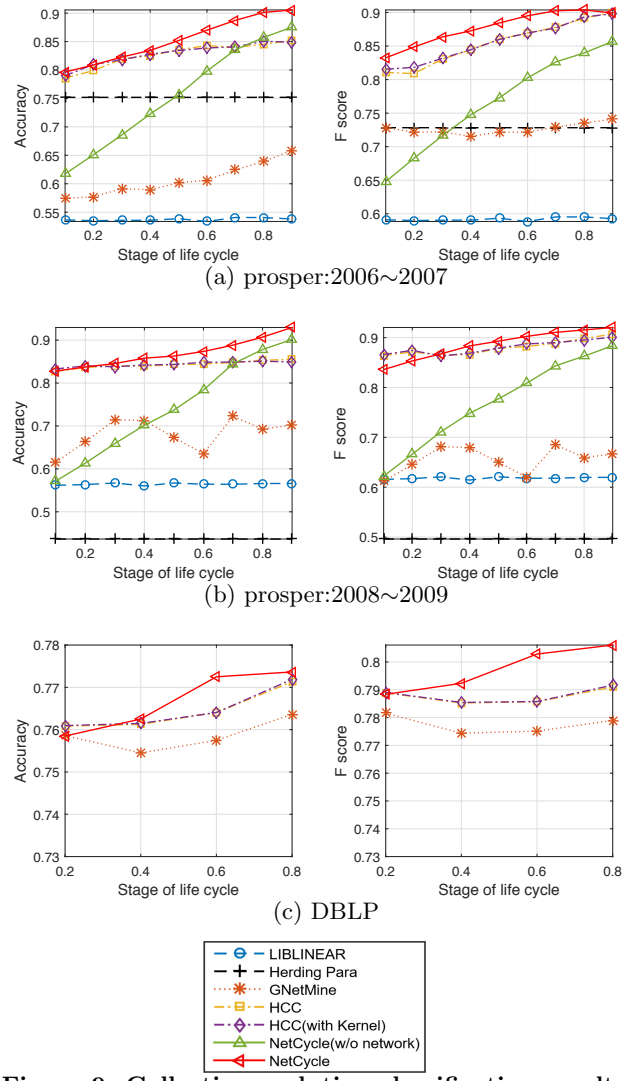
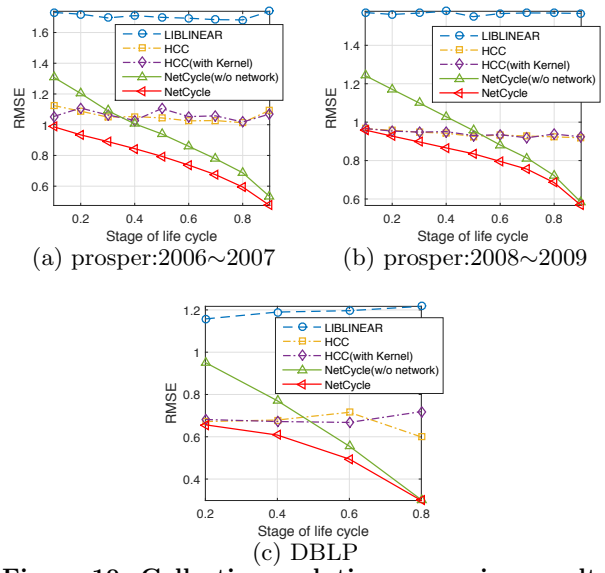


Figure 9: Collective evolution classification results.

4.3 Performances of Collective Evolution Classification

In our first experiment, we evaluate the effectiveness of the proposed NetCycle method on collective evolution classification problem. We apply our method on two problems: one is "biding detection" problem, which attempts to predict whether the loan listing will raise enough funds after fund-raising period. We binary the predicting result of fund-raising ratio to *Succeed* or *Failed* for comparison. Another is "academic career" problem, which tries to predict whether one author will engage in academic after graduation. The task can be represented as whether the author will publish his/her paper after five years. By hiding the response variables and links after stage k, we test all algorithms since different beginning stage k, and predict the final response values after their life cycles. Note that people pay more attention to positive instances, *e.g.*, investor only care about the loan listings which can receive enough funds in the end, and people care more about whether the author will continue to engage in academic career after PhD. We add F_β score as another classification evaluation metric, here we set $\beta = 0.5$. The results are reported in Figure 9. Due to the accuracy and F score are too small of logistic regression and



(a) prosper:2006∼2007 (b) prosper:2008∼2009

(c) DBLP

Figure 10: Collective evolution regression results.

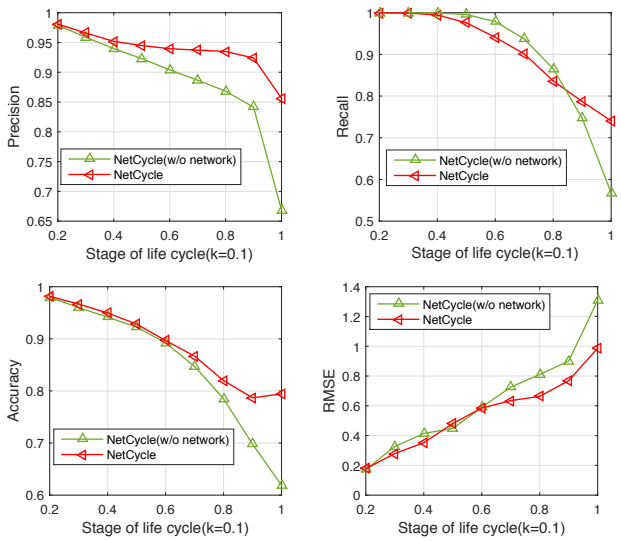


Figure 11: Node evolution during life cycle

NetCycle (w/o network) in DBLP dataset (less than 55%), so we did not shown the result in Figure 9(c).

The first observation in Figure 9 is: Most collective classification methods, *e.g.*, HCC and GNetMine, can achieve better accuracies than the baseline logistic regression. These results support that the heterogeneous dependencies among instances can improve classification performance. We also find that NetCycle (w/o network) and Herding Para are consistently and significantly outperforms than baseline, which support our claim that the evolution information of instances can also improve classification performance. Compared with all above algorithms, NetCycle always has best performance in each stage of life cycle, which explains that both heterogeneous dependencies among instances and the evolution information of node response variables can improve prediction performance in the classification task. Finally, we find that HCC method with kernel does not perform well, although it utilizes both heterogeneous dependencies among instances and the temporal information of instances. One possible reason is that it is hard to choose the suitable kernel to model the graph. Our algorithm almost achieves the best performance in all datasets.

4.4 Performances of Collective Evolution Regression

Compare with all above competing algorithms, NetCycle is not only able to solve the classification problem, but also able to solve the regression problem. In order to validate the effectiveness of NetCycle, we made slight modifications to the competing algorithms, by referring the works of [13] in collective regression. The competing algorithms include: LIBLINEAR, HCC, HCC (with Kernel) and NetCycle (w/o network). In this experiment, we apply NetCycle in collective evolution regression on two problems: predicting "how much fund of the loan will be raised after bidding period", and predicting "how many papers one author will publish during his/her PhD". The results are shown in Figure 10.

The first observation in Figure 10 we have is that most approaches have better performance than the baseline in each dataset, especially in latter stages of life cycle. We also find that although most algorithms' RMSEs have slightly decreasing tendency as time goes on, our methods (including w/o network) have more obviously decreasing tendency during life cycle, with the help of evolution information of the response variables during life cycles. Compared with NetCycle (w/o network), NetCycle have better performance in the early stage of life cycle, which demonstrates the heterogeneous dependencies among instances can improve performance in collective regression problem. In all datasets, NetCycle has outstanding performance in every stage.

4.5 Performances of Node Evolution Process Prediction

In our third experiment, we validate NetCycle (including w/o network) on predicting node evolution process in Prosper: 2006∼2007 dataset. We record the evolution process of the nodes since stage $k = 0.1$, and use following metrics to evaluate the performance in each stage during their life cycles, including: *precision* of positive nodes, *recall* of positive nodes, *accuracy* and *RMSE*. The results are shown in Figure 11, where similar trend holds for the other two datasets which cannot be shown due to space limitation.

We can find that both NetCycle and NetCycle (w/o network) have high precision in early stage, and only have little descent as time goes on. The decreasing tendency of NetCycle is greater than NetCycle (w/o network). The recall of positive nodes is also very high in the early stage, but the decreasing tendency is more obvious than precision for both algorithms. The results show that the decreasing performance in dataset mostly due to the descent of recall. The last two figures are accuracy and RMSE, where the errors are monotone increasing for both algorithms, however, NetCycle has much slower increment speed of errors than NetCycle (w/o network), which demonstrates our claim that the instances are collective evolution with life cycles.

4.6 Influence of different Meta paths

In this experiment, we study the influence of different meta paths on the collective evolution inference performance of our NetCycle model. Different types of meta paths have different semantic meanings, which correspond to different types of dependencies among the instances in HINs. We summarize all metapaths in Table 4, which are extracted

Table 4: Summary of Meta Paths among Loan Nodes

Notation	Meta Path	Semantics of the Dependency
LBL	$Loan \xrightarrow{created^{-1}} Borrower \xrightarrow{created} Loan$	Co-borrower relation
$LLeL$	$Loan \xrightarrow{bidding^{-1}} Lender \xrightarrow{bidding} Loan$	Co-investor relation
$LBBL$	$Loan \xrightarrow{created^{-1}} Borrower \xrightarrow{friend} Borrower \xrightarrow{created} Loan$	Loan's borrower are friend
$LLeLeL$	$Loan \xrightarrow{bidding^{-1}} Lender \xrightarrow{friend} Lender \xrightarrow{bidding} Loan$	Loan's lender are friend
$LBLeL$	$Loan \xrightarrow{created^{-1}} Borrower \xrightarrow{friend} Lender \xrightarrow{bidding} Loan$	Loan's borrower and Loan's lender are friend
$LLeBL$	$Loan \xrightarrow{bidding^{-1}} Lender \xrightarrow{friend} Borrower \xrightarrow{created} Loan$	Loan's lender and Loan's borrower are friend
$LBGBL$	$Loan \xrightarrow{created^{-1}} Borrower \xrightarrow{joined} Group \xrightarrow{joined^{-1}} Borrower \xrightarrow{created} Loan$	Loan's borrower joined same group
$LLeGLeL$	$Loan \xrightarrow{bidding^{-1}} Lender \xrightarrow{joined} Group \xrightarrow{joined^{-1}} Lender \xrightarrow{bidding} Loan$	Loan's lender joined same group
$LBGLeL$	$Loan \xrightarrow{created^{-1}} Borrower \xrightarrow{joined} Group \xrightarrow{joined^{-1}} Lender \xrightarrow{bidding} Loan$	Loan's borrower and lenders joined same group
$LLeGBL$	$Loan \xrightarrow{bidding^{-1}} Lender \xrightarrow{joined} Group \xrightarrow{joined^{-1}} Borrower \xrightarrow{created} Loan$	Loan's lenders and borrower joined same group

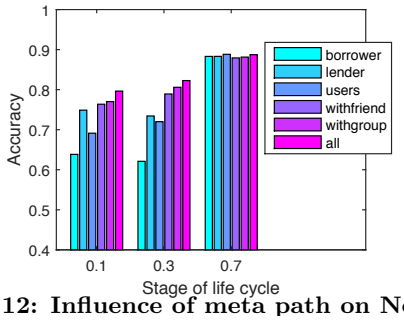


Figure 12: Influence of meta path on NetCycle.

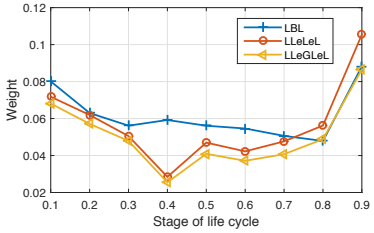


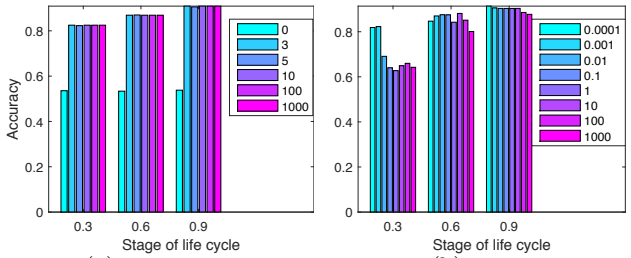
Figure 13: The weights of meta paths during life cycle.

from P2P lending network. In order to illustrate the influence of each path, we apply NetCycle with 6 groups of meta paths in the Prosper networks, including: *only borrowers*{LBL}, *only lenders*{LLeL}, *users*{LBL and LLeL}, *withfriend*{*users*, LBBL, LBLeL, LLeLeL and LLeBL}, *withgroup*{*users*, LBGBL, LBGLeL, LLeGLeL and LLeGBL} and *all*. We test each meta-path group in 3 stages of loan's life cycle in Prosper:2006∼2007 dataset, the results as shown in Figure 12. We find that NetCycle can get better performance with the increase number of meta-path types. This support our intuition that meta path is very expressive and can represent indirect relationships that are very important for collective evolution inference tasks.

4.7 Case Studies

To better understand the output of our methods, we give a case study of the weights of meta paths in each stage of life cycle, to show the effectiveness of NetCycle in matching the node evolution pattern. The meta paths we used are from Prosper:2006∼2007 dataset, including: *only borrowers* (LBL), *only lenders* (LLeLeL), and *with group* (LLeGLeL). In Figure 13, we display the weights of each meta path learned by segment models during the life cycle.

As shown in Figure 13, the first observation is that all meta paths have large weights in the early stages, and the weights of meta paths have a declining tendency in the early stages. This result proves a phenomenon: "investors prefer to invest new loan listings", which validated by Ceyhan [2]. The weights of meta paths *LLeLeL* and *LLeGLeL* have an similar increasing tendency in latter stage, it may be due to the following reason: as time goes on, the increased numbers of bids will attract more lenders to invest the loan, which also known as *herding effect* in financial domains. Ceyhan [2] also verified the herding effect in P2P Lending service. Compared with *LLeLeL* and *LLeGLeL*, we can find that the weights of *LBL* did not have obvious variation during loan's life cycle, because the relationship between borrower and loan will not change in the fund-raising period. There exists a sharp increasing tendency for all meta paths in the last stage, however, the weights of all features have a sharp variation in the last stage, because we need to normalize the results into discrete values in the last stage. These variation tendency of meta paths provides insightful knowledge for us to understand the node evolution process.



(a) Max_{iter} (b) λ

Figure 14: Influence of Parameters on NetCycle.

4.8 Model selection

There exist two essential parameters in NetCycle, Max_{iter} and λ. Max_{iter} is the maximum number of iterations # in the iterative inference part of NetCycle, and λ is the weight of LIBLINEAR's penalty item (L2 regularization). In previous experiments, we empirically set Max_{iter} as 10, and set λ as 0.001. To test the stability of the performances of NetCycle method, we test the accuracies of different values of Max_{iter} and λ in 3 stages of loan's life cycle on Prosper:2006∼2007 dataset, the result as shown in Figure 14. Similar trend holds for the other two datasets which cannot be shown due to space limitation.

Figure 14 illustrates that NetCycle performs quite well and stably, and it is not sensitive to the values of Max_{iter}

and λ. In Figure 14(a), when the number of iterations is very small, *e.g.*, Max_{iter}<3, the results did not perform well, because the algorithm not reached convergence yet. However, with the increase of the number of iterations, the results do not change obviously. In Figure 14(b), we can find that smaller λ always has better performance at the early stage (k=0.3), and the results don't change obviously as time goes on (*e.g.*, k=0.6 or 0.9). There does not exist obvious differences at each stage.

5. RELATED WORK

Our work on collective evolution inference is related to both collective inference and dynamic networks analysis. We will introduce recent related works of them in this section.

Collective inference [6] of relational data, including collective classification [1] and collective regression [13], have been investigated by many researchers. Basic collective classification problems focus on classification in homogeneous network [17]. Ji [7] studied a specialized classification problem on heterogeneous networks, where different types of nodes share a same set of label concepts. Kong [11] proposed a method called HCC based on meta-path to solve the collective classification problem on one-type nodes in heterogeneous networks. Loglisci [13] studied collective regression for handling autocorrelation of network data. All of them only paid attention to static networks, instead of dynamic networks. In this work, we compare with the methods in [7, 11] as two competing algorithms.

On the other hand, in dynamic networks, Sharan [19] proposed a representational framework to model both temporal and relational dependencies in networks, but they only tested the model on the link prediction problem. Yu [21] used cascading process to predict the propagation of information. Cho [3] presented a point process to model the Spatial-Temporal Networks. However, none of them focused on collective inference problem. Moreover, the evolution inference problem are very similar with autoregressive process. They both try to infer a time series values of the instances response variables in the future. But autoregressive process commonly studied the evolution pattern from previous sequence, while the evolution inference studied from correlated instances with the restrictions of their life cycles.

In addition, there exist several works in P2P lending networks [2, 14]. Ceyhan [2] examined a simple model to validate the lenders' bidding behavior during fund-raising period on bidding detection problem, which we also used as one of competing algorithms.

6. CONCLUSION

In this paper, we first present the collective evolution problem in dynamic networks, where the response variables of the nodes are evolving with a certain period time. Then we extend the problem into HINs, where the network includes multiple types of nodes connected through multiple types of links. We propose an effective algorithm, called NetCycle, to solve the collective evolution inference problem in HINs. The NetCycle method can not only predict the values of node response variables for collective inference problems, but also can predict the evolution tendency of the node response variables during life cycles. Empirical studies on real-world tasks demonstrate the effectiveness of the proposed method.

7. ACKNOWLEDGMENTS

The work was supported in part by the National High Technology Research and Development Program of China No.2015AA020105, and the National Natural Science Foundation of China Projects No.61170096, No.71331005. The authors would like to thank Jinhong Mi and Lu Ruan for their comments.

8. REFERENCES

[1] M. Bilgic, T. Eliassi-Rad, P. Sen, G. Namata, L. Getoor, and B. Gallagher. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[2] S. Ceyhan, X. Shi, and J. Leskovec. Dynamics of bidding in a P2P lending service: effects of herding and predicting loan success. In *WWW*, 2011.

[3] Y. S. Cho, A. Galstyan, J. Brantingham, and G. Tita. Latent point process models for spatial-temporal networks. *DCDS-B*, 2013.

[4] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: a library for large linear classification. *JMLR*, 9(12):1871–1874, 2008.

[5] J. Hamilton. *Time Series Analysis*. Princeton Univ. Press, 1994.

[6] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. *KDD*, 2004.

[7] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *KDD*, 2011.

[8] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *ECML*, 2010.

[9] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, 2001.

[10] X. Kong, B. Cao, and P. S. Yu. Multi-label classification by mining label and instance correlations from heterogeneous information networks. In *KDD*, 2013.

[11] X. Kong, P. S. Yu, Y. Ding, and D. J. Wild. Meta path-based collective classification in heterogeneous information networks. In *CIKM*, 2012.

[12] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. Dynammo: mining and summarization of coevolving sequences with missing values. In *KDD*, 2009.

[13] C. Loglisci, A. Appice, and D. Malerba. Collective regression for handling autocorrelation of network data in a transductive setting. *JIIS*, 2015.

[14] C.-T. Lu, S. Xie, X. Kong, and P. S. Yu. Inferring the impacts of social media on crowdfunding. In *WSDM*, 2014.

[15] Q. Lu and L. Getoor. Link-based classification. In *ICML*, 2003.

[16] L. K. Mcdowell and D. W. Aha. Labels or attributes?: rethinking the neighbors for collective classification in sparsely-labeled networks. In *CIKM*, 2013.

[17] J. Neville and D. Jensen. Iterative classification in relational data. *In AAAI Workshop*, 2000.

[18] J. Ren, J. Wang, M. Li, H. Wang, and B. Liu. *Prediction of Essential Proteins by Integration of PPI Network Topology and Protein Complexes Information*. Springer Berlin Heidelberg, 2011.

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

[20] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. In *VLDB*, 2011.

[21] L. Yu, P. Cui, F. Wang, C. Song, and S. Yang. From Micro to Macro: Uncovering and predicting information cascading process with behavioral dynamics. In *ICDM*, 2015.