Predicting Neighbor Distribution in Heterogeneous Information Networks

Yuchi Ma* Ning Yang† Chuan Li‡ Lei Zhang§ Philip S. Yu¶‖

Abstract

Recently, considerable attention has been devoted to the prediction problems arising from heterogeneous information networks. In this paper, we present a new prediction task, Neighbor Distribution Prediction (NDP), which aims at predicting the distribution of the labels on neighbors of a given node and is valuable for many different applications in heterogeneous information networks. The challenges of NDP mainly come from three aspects: the infinity of the state space of a neighbor distribution, the sparsity of available data, and how to fairly evaluate the predictions. To address these challenges, we first propose an Evolution Factor Model (EFM) for NDP, which utilizes two new structures proposed in this paper, i.e. Neighbor Distribution Vector (NDV) to represent the state of a given node's neighbors, and Neighbor Label Evolution Matrix (NLEM) to capture the dynamics of a neighbor distribution, respectively. We further propose a learning algorithm for Evolution Factor Model. To overcome the problem of data sparsity, the learning algorithm first clusters all the nodes and learns an NLEM for each cluster instead of for each node. For fairly evaluating the predicting results, we propose a new metric: Virtual Accuracy (VA), which takes into consideration both the absolute accuracy and the predictability of a node. Extensive experiments conducted on three real datasets from different domains validate the effectiveness of our proposed model EFM and metric VA.

*Sichuan University, Chengdu, China. scu.Richard.Ma@gmail.com

†Corresponding author. Sichuan University, Chengdu, China. yangning@scu.edu.cn

‡Sichuan University, Chengdu, China. lcharles@scu.edu.cn

§Sichuan University, Chengdu, China. leizhang@scu.edu.cn

¶University of Illinois at Chicago, Chicago, USA. psyu@uic.edu

‖Institute for Data Science, Tsinghua University, Beijing, China.

1 Introduction

As part of the recent surge of research on information networks, considerable attention has been devoted to prediction problems in heterogeneous information networks. The existing researches, however, mainly focus just on the predictions around a single link. For example, some works are interested in predicting whether or when a link will be built in the future [4, 6, 7, 12, 14, 16], and some works concern predicting strength of a link, such as predicting the ratings that customers will give to items or locations [2, 3, 9, 11]. Existing researches surprisingly pay little attention to the prediction of neighbor distributions, where states of neighbors are considered as a whole.

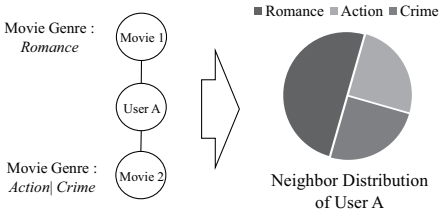


Figure 1: Neighbors Distribution

Fig. 1 offers an illustration of neighbor distribution. The left part of Fig. 1 illustrates *User A* has two movie nodes as its neighbors, which means *User A* rented two movies. The neighbor distribution of node *User A* is the distribution of labels on its neighbor nodes, as shown by the right part of Fig. 1, where the neighbors of a given node have equal weight, and the weight of a neighbor node is uniformly divided by its labels.

The neighbor distribution of a node usually evolves over time. For example, a user might rent movies of different genres as his/her taste changes. Such evo-

lution makes the prediction of neighbor distributions valuable for many different applications.

Motivating Example For an online sports video provider, the type distribution of subscribers is crucial to develop its sales strategy. The provider may be misled, if it only takes the recent sales data into consideration, and ignores the evolutionary feature of the type distribution. For example, the soccer fans are the major subscribers on May 2014, which may lead the provider to put more soccer advertisements online. However, the soccer fans are increasing slowly on May, and become the major subscribers on June 2014, as the opening of four-yearly soccer celebration "World Cup". Traditional recommender system methods may ignore the tiny increase of soccer fans on May.

In this paper, we aim at the problem of predicting the neighbor distribution of a given node in a heterogeneous information network, which has three main challenges we have to overcome:

• **Infinite state space of neighbor distributions** Since the fraction of a label is a real value, the number of possible states of a neighbor distribution is theoretically infinite. Traditional temporal models such as Markov chain cannot serve our goal because they often assume a finite state space.

• **Sparsity of heterogeneous links** In most cases, the links between one specific node and its heterogeneous neighbors are relatively sparse compared with the huge volume of a whole data set, e.g., "publishing" in DBLP, "rating" in Netflix and "checking in" in Foursquare. The sparsity of links between heterogeneous nodes makes it harder to mine sufficient meaningful patterns for individuals.

• **Fairly evaluating predictions** Not all nodes are equally predictable, hence the traditional metrics that just take absolute accuracy into account are unable to appropriately assess the predictions for the nodes that are less predictable. We need a new metric that can treat every node fairly.

In this paper, inspired by the idea of Factor Model [17, 18], we propose an Evolution Factor Model (EFM) to accurately predict neighbor distributions from the sparse data. Our main contributions can be summarized as follows:

(1) We introduce an Evolution Factor Model (EFM) for accurately predicting neighbor distributions. EFM employs our proposed data structures, Neighbor Distribution Vector (NDV) and Neighbor Label Evolution Matrix (NLEM), to represent the infinite state space of a neighbor distribution and capture the evolution of neighbor distributions respectively.

(2) We propose a new prediction metric, Virtual Accuracy (VA), which takes into consideration both the absolute accuracy and the difficulty of a prediction to fairly evaluate the prediction results of nodes with different predictabilities.

(3) We conduct extensive experiments on three real datasets, and compare EFM with an empirical method and two existing methods. The results validate the performance of our proposed model, algorithm and accuracy metric.

The rest of this paper is organized as follows. We give the problem definition and formalization in Section 2. In Section 3, we describe our prediction model EFM, and further present the learning algorithm for EFM. We discuss the predictability of nodes and propose a prediction metric in Section 4. We present the experimental results and analysis in Section 5. Finally, we discuss related works in Section 6, and conclude in Section 7.

2 Problem Definition

2.1 Heterogeneous Information Network

A heterogeneous information network contains multiple types of objects and links. In this paper, we only consider those heterogeneous information networks with star network schema [13], i.e., links only exist between the center type of nodes as **target nodes**, and several other types of nodes as **attribute nodes**. For example, in Location Based Social Network, the target nodes are users, and the attribute nodes can be venues, ratings or tips.

We denote the heterogeneous information networks with star network schema by $G = \langle V, E \rangle$, where V is

the node set and E is the link set. We denote the target and attribute node set by $\mathcal{X} \subset V$ and $\mathcal{U} \subset V$ respectively. The nodes and the links in networks are being constructed and destructed over time. In order to capture the dynamics, we use time window, which is denoted by T, to capture the neighbor distribution with timeliness from dynamic networks. The node set and the attribute node set in T are denoted by V_T and $\mathcal{U}_T$. We use T_h, T_c and T_f to represent the historical, current and future time windows respectively.

2.2 Label Distribution Vector

Assuming the universal label set of a given attribute node set $\mathcal{U}$ is denoted by $\beta_{\mathcal{U}} = \left\{\beta_{\mathcal{U}}^{(1)}, ..., \beta_{\mathcal{U}}^{(i)}, ..., \beta_{\mathcal{U}}^{(n)}\right\}$, where $\beta_{\mathcal{U}}^{(i)}$ is a label, and n is the number of label types, the definition of label distribution vector is given as follow:

Definition 1. ***Label Distribution Vector (LDV)*** *For a given attribute node $u \in \mathcal{U}$, its LDV, $\overrightarrow{v}_u \in \mathbb{R}^n$, is defined as:*

$$\overrightarrow{v}_u = (v_u^{(1)}, ..., v_u^{(i)}, ..., v_u^{(n)}), \tag{1}$$

where n is the number of all attribute nodes' label types; $v_u^{(i)} = \frac{I_u(\beta_{\mathcal{U}}^{(i)})}{\sum_{k=1}^{n} I_u(\beta_{\mathcal{U}}^{(k)})}$, $I_u(\beta_{\mathcal{U}}^{(i)})$ is 1 if u has the label $\beta_{\mathcal{U}}^{(i)}$, and 0, otherwise.

LDV measures the label distribution of an attribute node, which is fixed. For example, the labels of an article are subject areas, thus the LDV depicts the direction of this article.

2.3 Neighbor Distribution Vector

Definition 2. ***Neighbor Distribution Vector (NDV)*** *For a given target node $x \in \mathcal{X}$, the NDV of x's attribute node neighbors $\mathcal{U}'_T \subseteq \mathcal{U}_T$ in time window T, $\overrightarrow{w}_x(\mathcal{U}'_T) \in \mathbb{R}^n$, is defined as:*

$$\overrightarrow{w}_x(\mathcal{U}'_T) = (w_x^{(1)}, ..., w_x^{(n)}), \tag{2}$$

where n is the number of the given attribute nodes' label types; $w_x^{(i)} = \frac{\sum_{u \in \mathcal{U}'_T}(v_u^{(i)})+1}{|\mathcal{U}'_T|+n}, i \in [1, n]$.

Hereinafter, we just denote a NDV by $\overrightarrow{w}_x$ if the context is unambiguous. Note that: (1) For smoothing, we add 1 in the numerator and n in the denominator of $w_x^{(i)}$. (2) A node has an NDV corresponding to each different type of attribute node neighbors. For example, in DBLP, there are two type of attribute nodes, which are "Articles" and "Journals"; therefore, the target node "Scholar" should have two NDVs, one for its "Article" neighbors and the other for its "Journal" neighbors.

2.4 Problem Statement

Based on NDV, we can formally state the problem of Neighbor Distribution Prediction (NDP) as follow :

Assigned the historical, current and future time window T_h, T_c and T_f respectively, given a target node x and its NDV of x's attribute node neighbors $\mathcal{U}$ in time window T_h and T_c, we want to predict $\overrightarrow{w}_x(\mathcal{U}'_{T_f})$.

3 Evolution Factor Model

In this section, we describe our Evolution Factor Model (EFM). At first, we briefly introduce the basic idea of Factor Model.

3.1 Evolution Factor Model

Recommender system methods based on latent factor matrix model take the data in historical and current time window as a whole, but dismiss the evolution of the network. In order to capture the dynamics, Evolution Factor Model first stores the probability of changes from one label to another, which leads to the following definition of Neighbor Label Evolution Matrix (NLEM).

Definition 3. ***Neighbor Label Evolution Matrix*** *For a given node x, its Neighbor Label Evolution Matrix of attribute nodes $\mathcal{U}$ from time window T_p to T_q, denoted by $L_x^{<\mathcal{U}'_{T_p}, \mathcal{U}'_{T_q}>} \in \mathbb{R}^{n \times n}$, is a matrix in which a cell $L_x(i, j)$ is the probability that x's neighbor label changes from $\beta^{(j)}$ to $\beta^{(i)}$, i.e.,*

$$L_x(i, j) = P(\beta^{(i)} | \beta^{(j)}). \tag{3}$$

Evolution Factor Model Based on NLEM, for a given target node x, we can predict its NDV $\overrightarrow{w}_x(\mathcal{U}'_{T_f})$ as the transformation of its historical NDV, $\overrightarrow{w}_x(\mathcal{U}'_{T_h+T_c})$ through its NLEM $L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>}$, which leads to our EFM as follows:

$$\overrightarrow{w}_x(\mathcal{U}'_{T_f}) = L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>} \times \overrightarrow{w}_x(\mathcal{U}'_{T_h+T_c}). \tag{4}$$

Note that the essence of the matrix product in EFM is different from that in a general factor model. In a general factor model, the matrix product is static, which consider the data in historical and current time window as a whole. In contrast, NLEM, in our EFM, which can capture the changes from historical time window to current time window agilely, and changes as the given target node's neighbor labels evolve over time.

3.2 Model Learning

To overcome the issue of data sparsity, in the light of the heuristic knowledge that similar individuals have similar behaviors, we first apply a clustering algorithm to all the target nodes, and learn the NLEM for the cluster which x belongs to. According to this idea, given target node x and the node set $\mathcal{X}'$ consisting of the nodes belonging to the same cluster of x, we can learn NLEM as follow:

$$L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>} = \underset{L}{argmin} \sum_{x' \in \mathcal{X}'} \epsilon^2, \tag{5}$$

where $\epsilon = |\widehat{\overrightarrow{w}}_{x'}(\mathcal{U}'_{T_c}) - \overrightarrow{w}_{x'}(\mathcal{U}'_{T_c})|$, $L \in \mathbb{R}^{n \times n}$, $\widehat{\overrightarrow{w}}_{x'}(\mathcal{U}'_{T_c}) = L \times \overrightarrow{w}_{x'}(\mathcal{U}'_{T_h})$.

Note that $\widehat{\overrightarrow{w}}_{x'}(\mathcal{U}'_{T_c})$ is an estimate of $\overrightarrow{w}_{x'}(\mathcal{U}'_{T_c})$ for target node x'. So NLEM is actually defined as the optimal matrix that minimizes the overall error of the estimates over all target nodes in $\mathcal{X}$. Thus, for learning NLEM, we adopt least square method to establish linear regression, indicating the NDVs in T_h and T_c as follows:

$$L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>} = (X^T X)^{-1} X^T Y, \tag{6}$$

where

$$X = \begin{Bmatrix} X^{(1)} = \overrightarrow{w}_{x^{(1)}}(\mathcal{U}'_{T_h}) \\ \cdots \\ X^{(N)} = \overrightarrow{w}_{x^{(N)}}(\mathcal{U}'_{T_h}) \end{Bmatrix},$$

$$Y = \begin{Bmatrix} Y^{(1)} = \overrightarrow{w}_{x^{(1)}}(\mathcal{U}'_{T_c}) \\ \cdots \\ Y^{(N)} = \overrightarrow{w}_{x^{(N)}}(\mathcal{U}'_{T_c}) \end{Bmatrix}.$$

The learning algorithm for EFM is shown in Algorithm 1.

Algorithm 1 *Learning Algorithm for EFM* $(x, T_h, T_c, \mathcal{X}_{T_h}, \mathcal{X}_{T_c}, \mathcal{U}, K)$

INPUT:

x: A given node;
T_h: Assigned historical time window;
T_c: Assigned current time window;
$\mathcal{X}_{T_h}$: A subset of $\mathcal{X}$ in time window T_h;
$\mathcal{X}_{T_c}$: A subset of $\mathcal{X}$ in time window T_c;
$\mathcal{U}$: The given attribute node set;
K: The parameter of K-means;

OUTPUT:

$L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>}$: The NLEM needed to be learned;

$\mathcal{X}' = \Phi$, $X = \mathbf{0}$, $Y = \mathbf{0}$, W $= \{x' \,|\, x' \in \mathcal{X}_{T_h}, \mathcal{X}_{T_c}\}$;
for each $x' \in$ W **do**
  Compute $\overrightarrow{w}_{x'}(\mathcal{U}'_{T_h})$ and $\overrightarrow{w}_{x'}(\mathcal{U}'_{T_c})$;
end for
Do $K-$means on W based on the distance between $\overrightarrow{w}_{x'}(\mathcal{U}'_{T_h})$;
$\mathcal{X}' = \{$the nodes of the cluster that x belongs to$\}$;
for $i = 1$ to $|\mathcal{X}'|$ **do**
  $X^{(i)} = \overrightarrow{w}_{x^{(i)}}(\mathcal{U}'_{T_h})$;
  $Y^{(i)} = \overrightarrow{w}_{x^{(i)}}(\mathcal{U}'_{T_c})$;
end for
Compute $L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>}$;

In our learning algorithm, any classical clustering algorithm is qualified for our learning algorithm. We choose K-means as the clustering algorithm, and the selection of K is discussed in Section V.

3.3 Prediction

Given a target node x, its NDV $\overrightarrow{w}_x(\mathcal{U}'_{T_h+T_c})$ and the learned NLEM $L_x^{<\mathcal{U}'_{T_h},\mathcal{U}'_{T_c}>}$, the prediction of $\overrightarrow{w}_x(\mathcal{U}'_{T_f})$ can be made by EFM(Equation (4)).

4 Prediction Metric

4.1 Normalized Absolute Accuracy

We can intuitively measure the absolute accuracy of predictions for the given node x in terms of the Euclidean distance between a true $\overrightarrow{w}_x$ and its estimate $\widehat{\overrightarrow{w}}_x$.

By Definition 2, components of an NDV are positive and the sum of them is equal to 1, so the Euclidean distance between any two NDVs is less than or equal to $\sqrt{2}$. Then we can define the normalised absolute accuracy as follow:

$$\eta_x = 1 - \frac{d(\widehat{\overrightarrow{w}}_x, \overrightarrow{w}_x)}{\sqrt{2}}, \tag{7}$$

where $d(\widehat{\overrightarrow{w}}_x, \overrightarrow{w}_x)$ is the Euclidean distance between $\widehat{\overrightarrow{w}}_x$ and $\overrightarrow{w}_x$.

4.2 Predictability

As we have mentioned, it is unfair to assess a prediction just in terms of absolute accuracy, since the predictability of nodes are different.

Intuitively, the predictability of a node is relevant to its susceptibility to the similar homogeneous nodes. Specifically, the easier the node can be influenced by others, the more disordered its temporal pattern is, and the greater its predictability is. For example, in a given research field, the leading scholars' directions are difficult to capture, because they change their research directions rarely and such changes are mainly breakthroughs. These changes can hardly be predicted compared with their long-term stable studies. In contrast, the research direction of a PhD candidate is more likely influenced by his/her supervisor or the leading scholars. Inspired by this observation, we can define Prediction Difficulty as the measure on how difficult to predict a given node's NDV.

Definition 4. ***Prediction Difficulty (PD)*** *For a node x, the prediction difficulty of its NDV of attribute node neighbors $\mathcal{U}'_T$ in time window T, denoted by $g_x(\mathcal{U}'_T)$ is defined as:*

$$g_x(\mathcal{U}'_T) = 1 - h_x(\mathcal{U}'_T)/2,$$

where $h_x(\mathcal{U}'_T)$ is the temporal entropy of x's NDV in time window T, and

$$h_x(\mathcal{U}'_T) = -\sum_{i=1}^{n} w_x^{(i)}(\mathcal{U}'_T) log_n w_x^{(i)}(\mathcal{U}'_T).$$

Note that, $w^{(i)} \in (0,1)$, so $h_x(\mathcal{U}'_T) > 0$, and when $w^{(i)} = 1/n, \forall i \in [1,n]$, $h_x(\mathcal{U}'_T)$ reaches the maximum, which equals $-\sum_{i=1}^{n} \frac{1}{n} log_n \frac{1}{n} = 1$. Thus, $h_x(\mathcal{U}'_T) \in (0,1]$ and $g_x(\mathcal{U}'_T) \in [1/2, 1)$. In Definition 4, we use temporal entropy $h_x(\mathcal{U}'_T)$ to measure how disordered a node's temporal pattern is. We can see the more disordered the temporal pattern, the greater the temporal entropy, and consequently the less the prediction difficulty, which is in line with our expectation.

4.3 Virtual Accuracy

Now we further define Virtual Accuracy based on absolute accuracy and prediction difficulty as follow:

Definition 5. ***Virtual Accuracy (VA)*** *For a prediction of $\overrightarrow{w}_x(\mathcal{U}, T_f)$, its Virtual Accuracy, denoted by δ_x, is defined as:*

$$\delta_x = \eta_x \times g_x, \tag{8}$$

where η_x is the absolute accuracy and g_x is the prediction difficulty.

As Equation (8) shows, we define VA of a prediction as the product of the absolute accuracy and the predictability of that prediction. Since η_x and g_x are both nonnegative, it is obvious that VA favors the predictions whose absolute accuracy and difficulty are both great. As we can see in later experiments, η_x is negatively correlated with g_x. Thus even the absolute accuracy of a difficult prediction is low, the VA of it can still be expected be not low since its prediction difficulty is large. On the other hand, even the absolute accuracy of an easy prediction is high, the VA of it is expected to be low due to its small prediction difficulty.

5 Experimental Evaluation

5.1 Datasets

We learn the NLEM from the NDVs in T_h and T_c. For predicting neighbor distribution, we take the NDVs in $T_h + T_c$ as training set, and the NDVs in T_f as test set.

The datasets we use to validate EFM and algorithm are from three different domains, DBLP (a Coauthor Network), Netflix (a Movie Rental Network), and Foursquare (a Location Based Social Network).

DBLP [15] indexes more than about 230 million articles and contains massive links to home pages of computer scientists. The labels of "Article" contain 25 directions on Computer Science, thus an NDV of a "Scholar" node consists of 25 components. By assigned historical, current and future time window $T_h = [2006, 2010)$, $T_c = [2010, 2011)$ and $T_f = [2011, 2012]$, we randomly select 1000 scholars who published articles in all the three time window.

Netflix [1] contains about more than 100 million rating records from about $480,000$ customers over about $17,000$ movie titles. The labels of "Movie" contain 28 genres crawled from the website IMDb [5], thus an NDV of a "User" node consists of 28 components. By assigned historical, current and future time window T_h = [Apr.12^{th} 2005, Oct.12^{th} 2005), T_c = [Oct.12^{th} 2005, Nov.12^{th} 2005) and T_f = [Nov.12^{th} 2005, Dec.12^{th} 2005], we randomly select $1,000$ users who have movie rating records in all the three time window.

Foursquare[2] involves about 4.3 million friendships and about $80,000$ check-in tips of users during 3 years. The labels of "Venue" contain 8 categories given by Foursquare. By assigned historical, current and future time window T_h = [0^{th} day, 966^{th} day), T_c = [966^{th} day, 996^{th} day) and T_f = [996^{th} day, 1026^{th} day], we randomly select 500 users who have check-in records during in all the three time window.

5.2 Baseline

In order to demonstrate the effectiveness of our EFM, we compare our method with the following baseline methods:

- **MVM** (Mean Value Method) MVM is an empirical method. It takes the mean of the latest NDVs of the nodes in the cluster that the predicted node belongs to, as the estimate of the next NDV of a given node.

- **MF** (Basic Matrix Factorization)[17] MF is proposed by B. Webb to solve the movie recommender problem in Netflix Price. MF assumes the features of objects can be expressed as a series of factors, and different types of objects have factors with the same amount. When predicting the preference of the given objects of type A for the objects of type B, the preferences (which is called "ratings" in many cases) can be expressed as the product of the factors of the given objects of type A and B. The general expression of factor model is:

$$R = PQ^T, \tag{9}$$

  where $P \in \mathbb{R}^{N \times D}$ is the factor matrix of the objects of type A. $Q \in \mathbb{R}^{M \times D}$ is the factor matrix of objects of type B. N and M are the number of the objects of type A and the number of the objects of type B, respectively. D is the factor number.

- **BiasedMF** (Biased Matrix Factorization)[9] BiasedMF is proposed by Paterek, which is an extension of Basic Matrix Factorization. BiasedMF adds biased rates to the objects of either type. The prediction formula is:

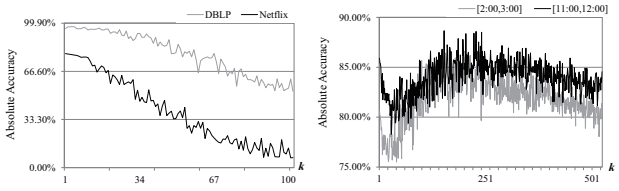
$$\hat{r}_{u,m} = b_u + b_m + \sum_{k=1}^{n} p_{u,k} \cdot q_{m,k}, \tag{10}$$

  where $\hat{r}_{u,m}$ is an estimate of rate that the object u of type A gives to the object m of type B. The $p_{u,k}$ and $q_{m,k}$ are the cells of the factor matrixes of type A and type B respectively. b_u and b_m are the biases of object u and m respectively.

In our experiments, we set the parameters of MF and BiasedMF as learning rate $\eta = 0.001$ and punishing parameter $\lambda = 0.02$, as suggested by Paterek [9] and Gorrell et al. [3]. We choose the number of NDV components as the latent feature numbers in MF and BiasedMF. Thus the feature numbers of DBLP, Netfilix and Foursquare are $25, 28$ and 8 respectively.

5.3 The Determination of K

Learning Algorithm for EFM requires the number of clusters, K, as the input when it invokes a K-means procedure, so we have to determine K before we start our experiments. For each dataset, we first randomly select 500 nodes from it, then apply our model to make predictions for these nodes and choose the K that maximizes the average absolute accuracy of the predictions. As Fig. 2 shows, we finally get $K = 5$ for DBLP, $K = 1$ for Netflix, and $K = 155, 156$ for Foursquare during $[2:00, 3:00]$ and $[11:00, 12:00]$ respectively.



(a) DBLP and Netflix (b) Foursquare

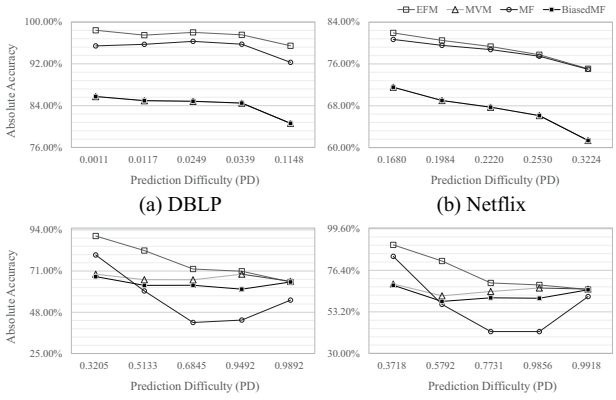
Figure 2: The Selection of K

5.4 The Validation of Predictability

Now we investigate how the absolute accuracy of a prediction correlates with its prediction difficulty. For each dataset, we first rank the nodes by PD in descending order, then divide the nodes into five groups. The nodes in a same group have equal PD. Finally we observe the absolute accuracies by applying EFM and three baseline methods, MVM, MF and BiasedMF, on the five groups respectively.

The results are shown in Fig. 3. We can see that the absolute accuracies of the methods we use in the experiments decrease in overall with the increase of the prediction difficulty. Such result validates our assumption that the more disordered the temporal pattern of a node is, the greater its predictability is. It also shows the necessity to assess a model by a fair metric which should take the predictability into consideration.

Note that, on Foursquare, the absolute accuracies of baseline methods do not decrease linearly with PD. It is because the human's daily routines are not all the same, which leads to the fluctuation of absolute accuracies of baseline methods. The absolute accuracy of EFM, however, has a linear decrease with PD. It is because EFM is not limited to the recognition of daily pattern, but instead takes the evolution regularity (represented by the neighbor label evolution matrix in EFM) into consideration. For example, the office workers who like nightlife can go to the nightclub for sleepover only on Weekends. For the nodes of that type, the two empirical methods can not perform as expected because the activities of sleepover on weekends are not common to everyone (which is the reason why MVM's curve fluctuates), or to an individual on everyday (which is the reason why the curve of MF and BiasedMF fluctuate).



(a) DBLP (b) Netflix

(c) Foursquare [2:00, 3:00] (d) Foursquare [11:00, 12:00]

Figure 3: The Relationship between Absolute Accuracy and Prediction Difficulty on Three Datasets

5.5 The Comparison between EFM and Baseline Methods

In this part, we compare the virtual accuracy of EFM with three baseline methods: MVM, MF and BiasedMF. The summarized result is shown in Fig. 4 and the detailed result is listed in Table 1. Our remarks on the result are as follows:

(1) EFM performs far better than MVM on all the datasets, while MVM has the worst performance

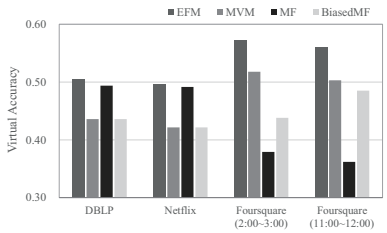


Figure 4: The Comparison of VA between EFM and Baseline Methods

Method	DBLP	Netflix	Foursquare [2:00, 3:00]	Foursquare [11:00, 12:00]
EFM	**0.5049**	**0.4960**	**0.5722**	**0.5606**
MVM	0.4359	0.4216	0.5179	0.5032
MF	0.4937	0.4917	0.3793	0.3619
BiasedMF	0.4360	0.4217	0.4381	0.4852

Table 1: Virtual Accuracies of EFM, MVM, MF and BiasedMF

on the Netflix and DBLP datasets.

(2) Although having the worst performance on the two Foursquare datasets, MF does have a good performance on Netflix dataset comparing with the other baseline methods, since MF is originally proposed for the movie recommendation problem in Netflix. EFM, however, still performs better than MF, which is because EFM can take into consideration not only the profile of the predicted nodes, but also the evolution regularity.

(3) As shown in Fig. 4, EFM outperforms BiasedMF, which performs similarly to MVM on DBLP and Netflix, but far worse on Foursquare.

(4) EFM outperforms MVM, MF and BiasedMF especially on Foursquare. This is because the three baseline methods only pay attention to the daily pattern (MVM) or the profile of users (MF and BiasedMF). However, the activities in Foursquare are limited to not only the daily pattern or profile of users, but also the evolution regularity over weeks, even months.

In summary, EFM is a robust and effective method. The VA of EFM is generally better than all the baseline methods.

6 Related Work

Three domains are relevant to our work, namely link prediction, rating prediction and factor model.

Link Prediction: Hasan et al. [4] first introduces supervised learning to predict whether a link will be built in the future. Wang et al. [16] introduces probabilistic model for link prediction. Leroy et al. [6] solves the cold start problem in link prediction. Lichtenwalter et al. [7] proposes new perspectives and methods in link prediction. Taskar et al. [14] propose a method to address the problem of link prediction in heterogeneous networks, based on the observations of the attributes of the objects. Sun et al. [12] extends the traditional link prediction to relationship prediction, which not only predicts whether it will happen, but also infers when it will happen. However, Sun's work, still focuses on the predictions of a single link.

Rating Prediction (Recommender System): Basically, the methods predicting ratings of links fall into two categories: memory-based algorithms and model-based algorithms. The memory-based algorithms directly make predictions based on homogeneous neighbors of a given node [8, 10, 19], while model-based algorithms make predictions based on a prediction model learned in advance. Savia et al.[11] proposes a prediction model based on bayesian networks. These existing methods pay insufficient attention to the evolution of neighbor distributions.

Factor Model: Factor Model assumes the features of objects can be expressed as a series of factors, which is also called Matrix Factorization (MF). MF is first proposed by Webb[17] to solve the movie recommender problem in Netflix Price. Based on Webb's work, G. Gorrell et al.[3] optimize the learning rate and the punishing parameter in MF. Paterek [9] proposes Biased Matrix Factorization (BiasedMF) to improve the performance of MF. However, the existing methods can not capture the dynamics of neighbor distributions agilely, which is exactly why we propose a new model EFM for our goal.

7 Conclusion

In this paper, we present a new prediction problem, Neighbor Distribution Prediction in heterogeneous information network. To address this problem, we propose an Evolution Factor Model (EFM), which takes Neighbor Label Evolution Matrix (NLEM) as the dynamic factor, and predicts the next NDV of a given node by transforming its current NDV by the NLEM. We also propose a learning algorithm for EFM, which learns the NLEM from the homogeneous nodes which are in the same cluster as a given node.

For fairly evaluating the predictions made by different methods, we propose Virtual Accuracy, which not only measures the absolute accuracy, but also takes the difficulty of a prediction into consideration.

We conduct the experiments on the datasets from three different applications, and compare EFM with three baseline methods: Mean Value Method, Basic Matrix Factorization and Biased Matrix Factorization. The results show EFM outperforms all the baseline methods in overall.

Acknowledgments

This work is supported by the National Science Foundation of China under Grant Nos. 61173099, 61103043 and the Doctoral Fund of Ministry of Education of China under Grant No. 20110181120062. This work is also supported in part by NSF through grants CNS-1115234, DBI-0960443, and OISE-1129076, and US Department of Army through grant W911NF-12-1-0066.

References

[1] Netflix prize. *URL http://www.netflixprize.com*.

[2] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. *GIS 2012*, 2012.

[3] G. Gorrell and B. Webb. Generalized hebbian algorithm for incremental latent semantic analysis. In *Proceedings of Interspeech*, 2006.

[4] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. *SDM' 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[5] IMDb. *URL http://www.imdb.com/*.

[6] V. Leroy, B. B. Cambazoglu, and F. Bonchi. Cold start link prediction. *KDD' 10*, 2010.

[7] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. *KDD' 10*, 2010.

[8] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. 2003.

[9] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filteringrecommendation algorithms. 2001.

[11] E. Savia, K. Puolamäki, and S. Kaski. Latent grouping models for user preference prediction. *Machine Learning*, 74(1):75–109, 2009.

[12] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla. When will it happen? - relationship prediction in heterogeneous information networks. *WSDM' 12*, pages 663–672, 2012.

[13] Y. Sun, J. Tang, J. Han, C. Chen, and M. Gupta. Co-evolution of multi-typed objects in dynamic star networks. *IEEE TKDE*, 2013.

[14] B. Taskar, M. fai Wong, P. Abbeel, and D. Koller. Link prediction in relational data. *NIP' 03*, 2003.

[15] D. Team. Dblp. *URL http://dblp.uni-trier.de/*.

[16] C. Wang, V. Satuluri, and S. Parthasarathy. Local probabilistic models for link prediction. *ICDM' 07*, pages 322–331, 2007.

[17] B. Webb. Netflix update: Try this at home. *URL http://sifter.org/simon/journal/20061211.html*.

[18] K. Yehuda, B. Robert, and V. Chris. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[19] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H. P. Kriegel. Probabilistic memorybased collaborative filtering. *IEEE TKDE*, 16(1):56–69, 2004.