Learning networks of heterogeneous information popularity

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

With the increasing popularity of the social networks and information-sharing platforms, many researches about network inference problem have been proposed to deal with such cases when we can only observe the activation time of nodes but the structure of networks are unavailable. In empirical analysis, we observe the fact that previous algorithms have long ignored the heterogeneity of diffusion speed in different life stages, which can result in unsatisfactory Network Inference results. We aim to improve the performance of inference by imposing some transplantable modification on existing method in which the extracted diffusion speed information is well used. But as the estimation of diffusion speed requires parent relationship that is seldom available, we propose a surrogate approach and prove its validity by calculating their correlation. An optimization method of averaging the life span to reduct noise in single cascade is introduced. Experiments on both synthetic dataset and real world dataset show significant improvement on inference accuracy when applying our method.

Introduction

Understanding the processes and dynamics of *information diffusion* through networks plays a fundamental role in a variety of domains, such as evaluating the effects of networks in marketing (Domingos and Richardson 2001; Kempe, Kleinberg, and Tardos 2003; Leskovec, Adamic, and Huberman 2007), monitoring the spread of news, opinions, and scientific ideas via citation networks (Adar et al. 2004; Gruhl et al. 2004; Leskovec, Kleinberg, and Faloutsos 2005), and detecting the spread of erroneous information (Dong, Berti-Equille, and Srivastava 2009).

In practical applications, the underlying diffusion network (e.g. networks on who influenced whom) is often hidden. Therefore, many interesting models have been developed to automatically infer diffusion networks from cascade observations, i.e., timestamps when users post a blog on certain topics or purchase products (Gomez-Rodriguez, Leskovec, and Krause 2012; Gomez-Rodriguez, Balduzzi, and Schölkopf 2011; Yang and Zha 2013; Zhou, Zha, and Song 2013; Wang et al. 2014; Daneshmand et al. 2014; Du et al. 2012; 2013).

To be more specific, the diffusion network is usually modeled as a matrix $ISA = \{\alpha_{u,v}\}$, where $\alpha_{u,v}$ is a parameter as the strength of influence user u has on v. Depending on the diffusion model, $\alpha_{u,v}$ can stand for the parameter of the delay distribution for information to propagate from u to v or the probability that v publish a related post given that u has published a post previously. The *Network Inference* problem focuses on discovering the all parameters $\alpha_{u,v}$ from the observed cascades.

Most previous work on network inference assume that the strength of influence $\alpha_{u,v}$ between u and v remains the same during the whole life of the diffusion processes. We can easily see that this assumption is unrealistic. Consider the propagation of a story in microblog as an example. It is more likely for a user to be influenced by friends to talk about the story when it is popular due to its freshness in the beginning of the propagation. On the contrary, when the story becomes stale and less popular towards the end of its life, it is less likely that one can influence her friends to have interest in it.

Ignoring the heterogeneity of influence strength in different stages can lead to unsatisfactory solution of the Network Inference problem. Assume that we observe user v posts the same contents immediately after u. Existing Network Inference algorithm will treat it as a strong evidence that user u has strong influence on v. However, it is also possible that v reposes fast simply because the story is very popular and any of her friends' post will lead to similar rapid response. On the other hand, if the above scenario occurs when the story is not no longer popular, v's rapid response to u's post can be safely treated as strong evidence that u is very influential on v. The confusion between strong influence between friends and the popularity of the story leads to the failure in this scenario.

In this work, we incorporate the heterogeneity of influence strength in different life stage of diffusion process to improve the accuracy of network inference. We design a heuristic referred to as the *Life Stage Heuristics* (LSH) that can be easily applied to any existing network inference algorithm. In our LSH, we model the strength of influence as a function of the diffusion life stage, namely $\alpha_{u,v}(t)$ instead of a constant $\alpha_{u,v}$ in previous network inference algorithms. We decouple the $\alpha_{u,v}(t)$ as a multiplication of two component, a time varying part $P(t)$ modeling the popularity of information and a constant part $\alpha_{u,v}$ modeling the strength of influence between pairs of individuals. Our formulation decomposes the true strength of influence from the confounding factor of popularity of the information. Moreover, the simplicity of the multiplicative form makes our heuristic easily applicable to any existing network inference algorithms to improve inference accuracy without changes in implementation nor increment in running time.

The rest of the paper is organized as follows. We first provide empirical evidence on the heterogeneity of influence strength and then provide a method to approximately estimate the popularity in different stages. We then show how our LSH can be applied to three sate-of-art network inference algorithms to improve the inference accuracy. We carry out extensive experiments on both synthetic and real world datasets to test the performance of our algorithms. Finally, we discuss related work and conclude the paper.

Empirical analysis

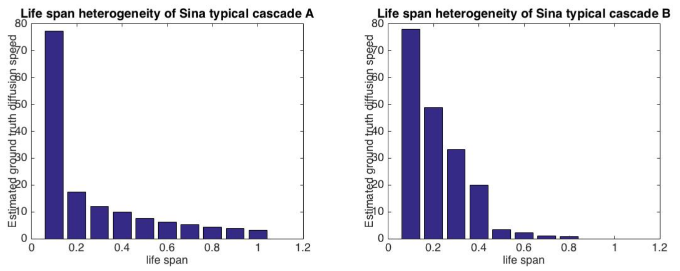
In this section, we carry out empirical analysis on two real world cascades datasets to show that the assumption of constant influence strength does not hold. We then provide a method to estimate the popularity of the information under propagation empirically from the observed cascades.

Existence of influence strength heterogeneity

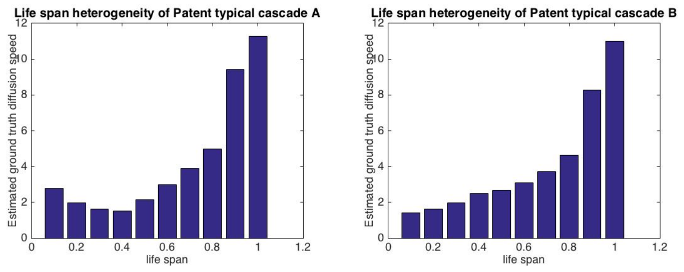
In many network inference models, the influence strength parameter $\alpha_{u,v}$ models the speed of diffusion, namely the time it takes for the information to propagation from user u to v. We empirically estimate the average diffusion speed as the inverse of delay time in two real world cascade datsets, Sina microblog and US Patent to show the heterogeneity of influence strength. [XH: Sina dataset contains the hyperlinks such as user u reposts one of v's microblog with delay $t_u - t_v$. Patent dataset contains citation links between patent products.]

The estimation process is as follows: we first normalize the span of each cascade to $[0, 1]$ and split the interval into ten bins. We treat each bin as one stage of the diffusion process. We estimate the average diffusion speed in each bin as the inverse of delay time $\frac{1}{\Delta_t}$ using the available explicit following information in the datasets. [XH: What is the unit of y-axis? Is it the diffusion speed after normalization? Ans: I think the inverse of diffusion speed is normalized but diffusion speed isn't. As the unit of y-axis contains normalization value, I'm confused about how to express the unit ∼∼] The results on two cascades from each dataset is shown in Figure 1.

The results clearly demonstrate the deviation from the assumption of constant diffusion speed. Another interesting observation is that the how the diffusion speed depends on the diffusion stages varies dramatically over different cascades and datasets. For the microblog datasets, usually the diffusion speed decrease when the story under propagation becomes stale and less interesting as time elapses. On the contrary, in the patent datsets, newly invented product gradually gains popularity leading to increment in the diffusion speed.



(a) Cascades of Sina Microblog dataset



(b) Cascades of U.S. Patent dataset

Figure 1: The diffusion speed in different stages of diffusion process in two real world datasets. [XH: Could we provide name for each cascade? link the story in microblog and the porduction in patent. Ans: I'm afraid there is no record in Sina dataset indicating the content of micro-blog; raw data of Patent contains the class of product, such as planting, Jewelry, Bridges, but there can be several different patent classes in a cascade, as one may cite to some patent products in other fields.]

Approximation popularity with number of activation

Direct estimation of average diffusion speed requires the diffusion pathway which is usually not available in real world datasets. It is even harder to estimate the average activation probability from observed cascades. As a result, we have to seek approximation for the true popularity for different stages in the diffusion process. In the work, we find the total number of activated nodes in each stage of the diffusion process as a good surrogate for the average diffusion speed or the activation probability as a measure for the popularity level.

Let $I^c(t)$ be the number of activated nodes in cascades c till time t. We approximate the popularity level with the following formula:

$$P^c(t) = (I^c(t + \frac{L}{2}) - I^c(t - \frac{L}{2}))/I^c(1).$$

That is to say, the estimated popularity level at time t is ratio of nodes activated in the windows of length L, $[t-\frac{L}{2}, t+\frac{L}{2}]$.

We carry out experiments to show that the proposed surrogate popularity level is a valid approximation. [XH: We sort all cascades descendingly by activation number and select the top 80 cascades in Patent dataset and top 50 cascades in Sina dataset]. For each cascade, we first normalize the time span to $[0, 1]$ and split it into K bins[XH: (we use $k = 10$

in this part)]. The true popularity level is assumed to be constant as the average diffusion speed estimated from the particular cascade. We compare the true popularity level and the surrogate popularity at each activation time t_a by computing the Pearson correlation for each cascade. The distribution on the correlation coefficient on the two datasets are shown in Table 1. More than 60% cascades and 80% cascades in Patent and Sina microblog dataset respective have correlation coefficient $r > 0.6$ with the true popularity level.

Table 1: Correlation coefficients between ground truth diffusion speed and the number of activated nodes in time unit.

coefficient range	strength	pct. Patent	pct. Sina
.00-.19	very weak	0.05	0.02
.20-.39	weak	0.125	0.02
.40-.59	moderate	0.175	0.06
.60-.79	strong	0.3	0.08
.80-1.0	very strong	0.35	0.82

Proposed algorithm

In this section, we'll introduce (C)LSH method that aims to improve inferring performance by taking advantage of information about messages' diffusion speed. Note that our methods don't ask for any extra input data other than traditional cascades set. We'll show that our method can be easily incorporated into existing network inference algorithms without adding any computational overhead. Some procedures for optimizing our method will be discussed at the end of this section.

Data and definitions

We can only observe the activation times t when diffusion happens. The first node being activated in node set N is called $root$. Node appeared after T is regarded as unactivated in this process. We call diffusion process as $cascade$, which contains the time stamp of activated nodes, $\mathbf{t}^c = (t_1^c, t_2^c, ...t_N^c)$. C is the collection of cascades, C=$\{\mathbf{t}^1, \mathbf{t}^2, ..., \mathbf{t}^{|C|}\}$.

Incorporation with existing algorithms

Network Inference algorithms always define the pairwise variable α_{ji} which is either the influence strength of edge or the probability that transmission happens on that edge. Previous algorithms assume that this variable of edge will not change in the whole life of diffusion which has been proved unreasonable in our empirical analysis. Here we combine the heterogeneity of influence strength with common Network Inference model by simply multiplying the popularity level $P^c(t)$ by transmission rate of edge, which is:

$$\hat{\alpha}_{ji}(t) = \alpha_{ji} * P^c(t); \tag{1}$$

Besides the common modification in Eq. 1, we'd like to show the special details when combining each previous algorithm:

- LSH-NetRate

We redefine the transmission likelihood function of Exponential, Power law and Rayleigh model with Eq. 1, which are:

$$f(t_i|t_j, \hat{\alpha}_{ji}) = \hat{\alpha}_{ji}(t_j) * e^{-\hat{\alpha}_{ji}(t_j)*(t_i-t_j)}$$
$$f(t_i|t_j, \hat{\alpha}_{ji}) = \frac{\hat{\alpha}_{ji}(t_j)}{\delta}(\frac{t_i-t_j}{\delta})^{-1-\hat{\alpha}_{ji}(t_j)}$$
$$f(t_i|t_j, \hat{\alpha}_{ji}) = \hat{\alpha}_{ji}(t_j)(t_i-t_j)e^{-\frac{1}{2}\hat{\alpha}_{ji}(t_j)(t_i-t_j)^2}$$

Note that range of $\hat{\alpha}_{ji}(t)$ is [0,1] too. Constraints of the optimization problem have no change.

- LSH-ConNie

The likelihood function in ConNie model is the product of 2 parts. The first part is about the probability of an infected node has at least one previously infected parent to activated it. The second part is the probability that no node ever infected an non-infected node. Variables $\hat{B}_{ji}^c = 1 - \hat{\alpha}_{ji}(t_j) = 1 - A_{ji} * P^c(t_j)$ and $\hat{\gamma}_c = 1 - \prod_{j:t_j \le t_i}(1 - w_{ji} * \hat{\alpha}_{ji}(t_j))$ are defined to transform the likelihood function into a convex optimization problem. When combining with LSH method, the amount of variables to be estimated increases as each cascade c needs a unique $\hat{B}$ matrix while all cascades share the same $\hat{B}$ in the original ConNie. We'd like to propose a compromise to reduce variables amount when cascades number is large. We skip the heterogeneity in the second part of likelihood function which indicates that $\hat{B}_{ji}^c$ can be redefined as $\hat{B}_{ji} = 1 - A_{ji}$. The number of variables reduces to $N * N + |C|$ as demanded.

- LSH-NetInf

In NetInf algorithm, $P_c(u, v)$ is the probability of transmission depending only on the time intervals. We redefine the transmission likelihood of edge considering both delay and popularity level, which is $\hat{\alpha}_{ji}(t_j) = P_c(u, v) * P^c(t_j)$[XH: duplicate name?~~]. The modified likelihood in LSH-NetInf of all cascades with the maximum weighted propagation tree is:

$$\begin{aligned} P(C|G) &= \prod_{c \in C} \sum_{T \in \mathcal{T}_c(G)} P(c|T) \\ &\approx \prod_{c \in C} \max_{T \in \mathcal{T}_c(G)} P(c|T) \\ &\approx \beta^q \varepsilon^{q'} (1-\varepsilon)^{s+s'} \prod_{(u,v) \in E_T} \hat{\alpha}_{ji}(t_j) \end{aligned}$$

As noted in NetInf paper, q is network edges number and q' is ε-edges number in T; s and s' refer to those that did not transmit respectively.

Optimization methods

Clustering cascades Extracting diffusion speed information on a single cascade is usually affected by noise as there are always some abnormal point. We use hierarchical clustering method as shown in Algorithm 1 to cluster similar cascades into a group. When computing popularity level, we use the average value at t_j in the same cascade group. By such means, we can reduce noises and keep the common properties of diffusion speed.

Algorithm 1 Clustering cascades with hierarchical clustering method

```
Require: number of clusters $k$; cascades set $C$
  for $c \in C$ do
    $G(c) \leftarrow$ c; // initialize each group with a single cascade
  end for;
  for $i = 1 : |C|$ do
    $Flag(i) \leftarrow i$; // initialize the group ID of each cascade
  end for;
  for each $c_1 \in C$ do
    for each $c_2 \in C$ do
      $D(c_1, c_2) \leftarrow$ Euclidean distance of $c_1's$ $S_p$ function
      and $c_2's$ $S_p$ function;
    end for;
  end for;
  while $|G| > 1$ do
    merge the nearest 2 clusters in $G$;
    update $D$ and $Flag$;
  end while
  return $Flag$;
```

Dealing with root node Each cascade has a root node who's time is fixed at 0, and this will increase $P^c(tj = 0)$ when we merged all cascades into one. This will affect every node as root's time is the earliest. We remove root node when calculating the activation number.

Experiments

Experiments for LSH and CLSH are performed to evaluate their abilities in improving network inference algorithms' performance. We mainly focus on 3 existing algorithms: NetRate, ConNie , NetInf, on which we apply LSH and CLSH method respectively.

Experiments on synthetic dataset

Synthetic networks of 256 nodes are generated from Kronecker model with parameter matrix [0.5 0.5; 0.5 0.5]. At the beginning of each cascade, root node is selected randomly and recorded as $t = 0$. Pairwise edge weights are sampled from $U(0, 1)$. The larger the value of edge weight, the higher the probability of transmitting a message between this pair of nodes. Once a node is infected, it begins to infect the adjacent nodes. Each activated node infects its neighbors with probabilities generated from transmission likelihood function(Exponential model). The diffusion proceeds iteratively until time window T, which is set at 10.

Measures in this paper are AUC(Area under Precision-Recall curve) and maximum F_1 score. F_1 score is the harmonic mean of precision and recall while we consider the maximum value in evaluation.

When generating synthetic cascade data, we use the piecewise function of $P^c(t)$ to control life span heterogeneity. Value of $P^c(t)$ implies the diffusion speed of message at t. Large deviation in $P^c(t)$ causes strong heterogeneity. In this part, $P^c(t)$ is set as [0.03 0.9 0.07 0 0], (time window $T = 10$ as noted) such as $P^c(t = 3.2) = 0.9$; $P^c(t = 5.9) = 0.07$.

Performance vs. cascade number. We make experiments of LSH-NetRate/LSH-ConNie/LSH-NetInf. By applying LSH method, the performance of network inference improves comparing with their original algorithms, as shown in Table 2 and 3. Figure 2 plots the Precision-Recall curve of LSH-NetRate/LSH-ConNie/LSH-NetInf method. Our methods make distinct improvement.

Table 2: Max F_1 score of LSH-NetRate/LSH-ConNie/LSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.6742	0.7913	0.8161
LSH-NetRate	0.7251	0.8210	0.8425
ConNie	0.7500	0.7526	0.7905
LSH-ConNie	0.7653	0.7734	0.8138
NetInf	0.5927	0.6799	0.7360
LSH-NetInf	0.6018	0.6997	0.7446

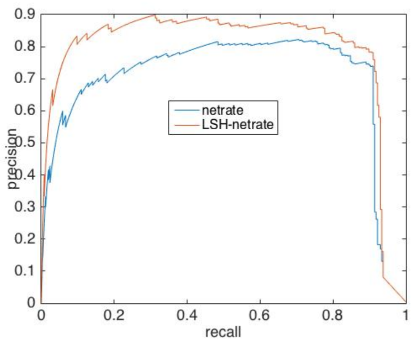
Table 3: Area under PR curve of LSH-NetRate/LSH-ConNie/LSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.4794	0.6299	0.6858
LSH-NetRate	0.5573	0.6892	0.7805
ConNie	0.6899	0.7424	0.7985
LSH-ConNie	0.6916	0.7657	0.8334
NetInf	0.3801	0.4638	0.5754
LSH-NetInf	0.4186	0.5421	0.5764

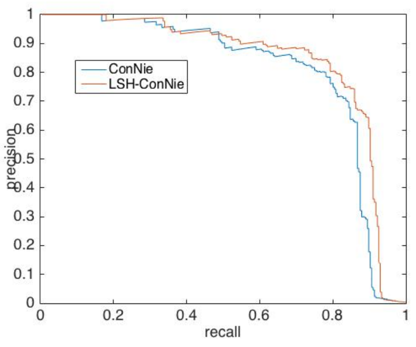
Performance vs. heterogeneity strength. We generate synthetic cascades with various heterogeneity strength, as shown in Table 4. The level of standard deviation depicts the strength of heterogeneity. Experiments in Figure 3 show that our method gains larger advantage to original NetRate as standard deviation of P^c increases.

Table 4: 4 types of Sp function with different standard deviation.

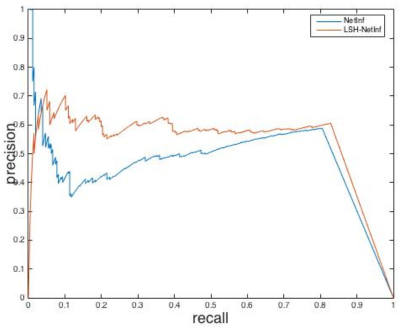
	P^c function	standard deviation
1	[0.2 0.2 0.2 0.2 0.2]	0
2	[0.2 0.4 0.3 0.05 0.05]	0.1541
3	[0.1 0.6 0.3 0 0]	0.2549
4	[0.03 0.9 0.07 0 0]	0.3924



(a) LSH-NetRate



(b) LSH-ConNie



(c) LSH-NetInf

Figure 2: Precision-Recall curve of LSH-NetRate/LSH-ConNie/LSH-NetInf method

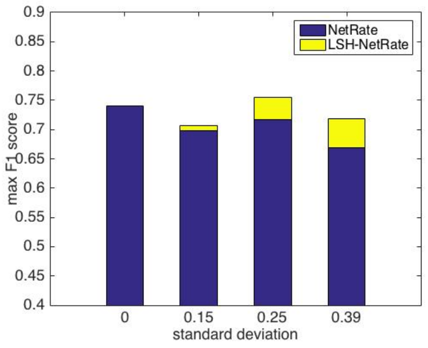


Figure 3: Experiments on different heterogeneity strength.

Setting of L. L is the length of sliding window around target time stamp that can affect the estimation of P^c. We run our methods with 4 different L settings separately, results are shown in Figure 4. We can see that their differences are slight as we change L's value. But LSH method performs better with L set around 10% of T.

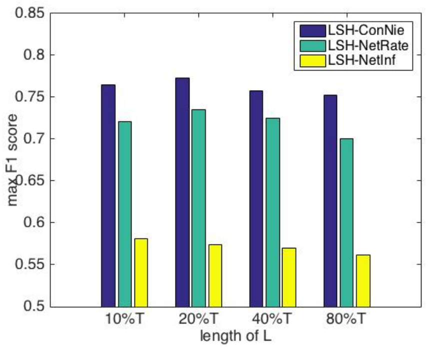


Figure 4: Experiments of different L values.

Experiments of CLSH method. CLSH method attempts to cluster cascades so that cascades in the same group have similar $P^c(t)$. In this part, we use 3 types of $P^c(t)$ functions: [0.1 0.8 0.1 0 0] / [0.8 0.2 0 0 0] / [0 0 0 0.2 0.8]. Before simulating each cascade, we randomly choose one of this three to be the specific P^c function of this cascade.

Performance of CLSH vs. cascade number. We make experiments of CLSH-NetRate/CLSH-ConNie/CLSH-NetInf methods on synthetic cascades with multiply heterogeneity functions. As shown in Table 5 and 6, our methods outperform the original algorithms on max F_1 score and AUC.

Table 5: Max F_1 score of CLSH-NetRate/CLSH-ConNie/CLSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.7124	0.7222	0.7254
CLSH-NetRate	0.7256	0.7425	0.7414
ConNie	0.7296	0.7531	0.7619
CLSH-ConNie	0.7474	0.7520	0.7692
NetInf	0.6727	0.7133	0.7316
CLSH-NetInf	0.6886	0.7314	0.7489

Table 6: Area under PR curve of CLSH-NetRate/CLSH-ConNie/CLSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.5243	0.5445	0.5894
CLSH-NetRate	0.5771	0.6527	0.6582
ConNie	0.7006	0.7449	0.7612
CLSH-ConNie	0.7194	0.7600	0.7602
NetInf	0.4079	0.4248	0.5138
CLSH-NetInf	0.4319	0.4402	0.5268

Setting of k. As shown in Figure 5, clustering cascades into $3 \sim 4$ groups produces higher accuracy.

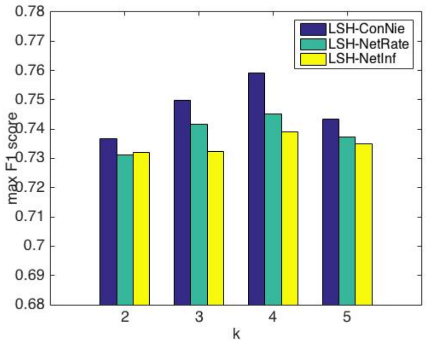


Figure 5: Experiments of different k values.

Experiments on real world dataset

We use *U.S. patent dataset* by *the National Bureau of Economic Research* to test (C)LSH method. The dataset spans 25 years from 1975 to 1999, containing 16,522,438 citations. Inventors of patents cite their sources. Therefore, we take inventors as the entities of network and follow the citations to reveal information flow. Note that we remove all self-loops in patent data, which happen when inventors cite their own previous works. By ranking inventors according to their activity, we have 147 most active inventors to form the ground truth network. We extract 150/250/350/450 transmission trees for cascade information. Experiments of (C)LSH-NetRate, (C)LSH-ConNie, (C)LSH-NetInf are made on this dataset.

For real world dataset, the lengths of cascades differ a lot. We apply Min-Max Normalization to activation times in each cascade separately. Cascades' length is 1 after transformation.

- (C)LSH-NetRate

  Performance vs. cascade number. From Table 7 we can see improvement by applying LSH method to NetRate on various data scale. Increasing k in some occasions can help improve accuracy.

Table 7: Real data: Max F_1 score of NetRate/ LSH-NetRate/ CLSH-NetRate vs. casNum

	cas 150	cas 250	cas 350	cas 450
NetRate	0.5915	0.6110	0.6418	0.6667
LSH-NetRate	0.6324	0.6364	0.6817	0.6975
CLSH-k 3	0.6328	0.6431	0.6731	0.6997
CLSH-k 5	0.6324	0.6360	0.6752	0.7121

- (C)LSH-ConNie

  Performance vs. cascade number. LSH method outperforms ConNie notably as shown in Table 8. Furthermore, applying CLSH to cluster similar cascades extends the performance advantage over original ConNie method.

Table 8: Real data: Max F_1 score of ConNie/ LSH-ConNie/ CLSH-ConNie vs. casNum

	cas 150	cas 250	cas 350	cas 450
ConNie	0.5702	0.6127	0.6412	0.6788
LSH-ConNie	0.8125	0.8222	0.8532	0.8673
CLSH-k 3	0.8249	0.8298	0.8502	0.8733
CLSH-k 5	0.8140	0.8218	0.8561	0.8581

- (C)LSH-NetInf

  Performance vs. cascade number. Table 9 and 10 present the AUC and maximum F_1 score results of our methods and NetInf algorithm. In all cases, our methods perform better than NetInf. When including more cascades, the complexity increases. Proper increment of k may lead to higher performance.

Table 9: Real data: Max F_1 score of NetInf/ LSH-NetInf/ CLSH-NetInf vs. casNum

	cas 150	cas 250	cas 350	cas 450
NetInf	0.5284	0.5382	0.5779	0.5981
LSH-NetInf	0.7604	0.7762	0.7993	0.7932
CLSH-k 3	0.7757	0.7902	0.8000	0.7938
CLSH-k 5	0.7833	0.7762	0.8129	0.8062

Table 10: Real data: AUC of NetInf/ LSH-NetInf/ CLSH-NetInf vs. casNum

	cas 150	cas 250	cas 350	cas 450
NetInf	0.7860	0.7919	0.8160	0.8292
LSH-NetInf	0.9413	0.9393	0.9410	0.9429
CLSH-k 3	0.9502	0.9473	0.9418	0.9436
CLSH-k 5	0.9547	0.9393	0.9490	0.9506

Conclusions

In this article, we design $Life\ Stage\ Heuristics$ method, called LSH, to improve the accuracy of Network Inference by making use of heterogeneous influence strength information. In contrast to previous Network Inference algorithms, such as NetRate, ConNie and NetInf, our method considers the fact that the popularity of a story(patent product, etc) changes with time and we propose an efficient approach to learn the popularity heterogeneous networks. Experiments on both synthetic dataset and real world dataset show that our method outperforms the original NetRate, ConNie, and NetInf models.

References

Adar, E.; Zhang, L.; Adamic, L.; and Lukose, R. 2004. Implicit structure and the dynamics of blogspace. In *Workshop on the Weblogging Ecosystem*, volume 13.

Daneshmand, H.; Gomez-Rodriguez, M.; Song, L.; and Schölkopf, B. 2014. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *Proc. 31st Intl. Conf. on Machine Learning*.

Domingos, P., and Richardson, M. 2001. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, 57–66.

Dong, X. L.; Berti-Equille, L.; and Srivastava, D. 2009. Truth discovery and copying detection in a dynamic world. *Proc. VLDB Endow.* 2(1).

Du, N.; Song, L.; Yuan, S.; and Smola, A. J. 2012. Learning networks of heterogeneous influence. In *Proc. 24th Advances in Neural Information Processing Systems*. 2780–2788.

Du, N.; Song, L.; Woo, H.; and Zha, H. 2013. Uncover topic-sensitive information diffusion networks. In *Proc. 16th Intl. Conf. on Artificial Intelligence and Statistics*.

Gomez-Rodriguez, M.; Balduzzi, D.; and Schölkopf, B. 2011. Uncovering the temporal dynamics of diffusion networks. In *Proc. 28th Intl. Conf. on Machine Learning*, 561–568.

Gomez-Rodriguez, M.; Leskovec, J.; and Krause, A. 2012. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data* 5(4).

Gruhl, D.; Guha, R.; Liben-Nowell, D.; and Tomkins, A. 2004. Information diffusion through blogspace. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, 491–501.

Kempe, D.; Kleinberg, J.; and Tardos, E. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, 137–146.

Leskovec, J.; Adamic, L. A.; and Huberman, B. A. 2007. The dynamics of viral marketing. *ACM Trans. Web* 1(1).

Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, 177–187.

Wang, S.; Hu, X.; Yu, P. S.; and Li, Z. 2014. Mmrate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *Proc. 20th Intl. Conf. on Knowledge Discovery and Data Mining*, 1246–1255.

Yang, S.-H., and Zha, H. 2013. Mixture of mutually exciting processes for viral diffusion. In *Proc. 30th Intl. Conf. on Machine Learning*.

Zhou, K.; Zha, H.; and Song, L. 2013. Learning social infectivity in sparse low-rank network using multi-dimensional hawkes processes. In *Proc. 30th Intl. Conf. on Machine Learning*.