# Link Weight Prediction in Signed Networks

**Shuyang Shi**[*‡], **Tong Yang**[*†], **Zhi Bie**[*†]
[‡] Dept. of Computer Science, Stanford University, CA, 94305
[†] Dept. of Electrical Engineering, Stanford University, CA, 94305
{bsnsk,tongy,zhib}@stanford.edu

## 1 Introduction

The Internet has witnessed an explosive growth of on-line communities. These communities often involve ratings between users, which can be modeled as weighted links between nodes (users) in a weighted signed network (WSN). A strong positive link may indicate the user trusts or likes the other user very much, while a weak negative link indicates the user distrusts or disagrees with the other user a little bit.

In this project, we explore the problem of predicting link weights in directed signed network. In order to capture the similarity of user behavior in rating other users, we propose to map node properties into vectors, and use inner product to estimate link weights. Inspired by collaborative filtering, we approximate the weight matrix of the network by matrix decomposition, and optimize through gradient descent. Experiments of predicting 1 link in network characterize the accuracy of the method, and experiments of predicting from 10% to 90% of links examine its robustness. By comparison with the state-of-art method, experiment results show that the proposed method outperforms other algorithms by a margin. The effectiveness is demonstrated on Bitcoin OTC, Bitcoin Alpha, and Wikipedia RfA datasets.

Besides, we explored two possible extensions of the model: sign correction and time factor. Firstly, based on the fact that sign prediction is easier than weight prediction, we can adopt some penalty term into cost function for correction, and possibly escalate the results of weight prediction. Secondly, the creation time of links, which is provided in Bitcoin datasets, can possibly influence weight prediction. We have two competing hypothesis: one is that if a link is created earlier, it has more influence on determining how much the rated node is trusted. The other is the recent ratings are more reliable than ratings in the past. These extensions are examined and analyzed through experiments.

The contributions of this work are

- proposing a new effective model for link weight prediction in WSNs,
- demonstrating the effectiveness of the model through experiments on real-world datasets and comparison with state-of-art models,
- testing possible extensions of the model by adding sign correction term or link creation time, and
- showing robustness of the model by varying link prediction size and network size.

## 2 Related Works

**Sign prediction in unweighted signed networks** Balance and status theory is widely used in predicting the sign of edges in unweighted signed networks. The theories were first proposed by Leskovec et al. [1] to study how the interplay between positive (friendly) and negative (antagonistic) relationships affects the stability of on-line social networks. Balance theory suggests that only triads with three positive signs and triads with one positive sign are stable. Status theory postulates that when person $A$ makes a positive link to person $B$, $A$ is asserting $B$ has higher status [2].

**Weight prediction in unsigned networks** Weighted unsigned networks contain only positive edges. Some methods involve some communication information between individuals [3]. Others use non-communication based techniques instead, which is more relevant to this project. Some existing algorithms are listed below.

- Reciprocal [4]: predicts weight for edge $(u, v)$ as the reciprocal of edge $(v, u)$ (0 if there is no reciprocal edge).

[*] Authors are listed alphabetically.

- Triadic balance [5]: extends balance theory, predicts weight for edge $(u, v)$ as the average product of edge weights for all incomplete triads that edge $(u, v)$ is a part of.
- Triadic status [6]: extends status theory, in a transitive triad, the predicted weight of the transitive edge is the sum of the two non-transitive edge weights.

**Weight prediction in signed networks** The first and very recent paper to explore weight prediction in weighted signed network(WSN) is [7], where Kumar et al. provided a way for link weight prediction in WSN by proposing two measures of node behavior, *goodness* (how much the node is liked / trusted) and *fairness* (how fair the node is in rating others). The formal definitions of goodness and fairness are, respectively,

$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f(u) \times W(u, v),$$

$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} \frac{|W(u, v) - g(v)|}{2},$$

where $W(u, v)$ is the weight of link $u \to v$, and $in(v), out(u)$ are the set of incoming and outgoing neighbors. These two measures can be obtained by iterations, and the prediction of a link from $u$ to $v$ is $f(u) \cdot g(v)$. The method exhibited the best effectiveness and robustness among different datasets and different extent of sparsity.

This method is effective and robust, but we believe using goodness and fairness (with only two directions of good or bad) for weight prediction can be limited. The link weights may depend more on the personal properties of members, and similarity between nodes can provide more insight.

## 3 Proposed Method

In this section we describe the proposed method inspired by the latent factor model. Firstly we explain some insights of mapping nodes to $K$-dimensional vectors, which can indicate link weights while maintaining some properties of similarities. Then we provide a definition to decompose the weight matrix into such vectors. At last, we also explore the time factor on link weight prediction.

### 3.1 Similarities by Vectors

As in previous work [7], we notice that the behavior of evaluating others and being evaluated are different even for one person. Unlike the fairness-goodness measures where the measurement is "how much" (i.e. large or small real number), we intend to map the measures to a $K$-dimensional space so that similarities can be implied.

Formally, for a node $u$ in the network, we find latent vectors

$$p_u \in \mathbb{R}^K, q_u \in \mathbb{R}^K,$$

to describe the characteristics of $u$ when evaluating others (i.e. links going from $u$), and the characteristics of $u$ when being evaluated (i.e. links coming to $u$). In this way, the nodes are mapped to $K$-dimensional spaces; the similarities between nodes can be evaluated by some properties of the vectors, for instance, the cosine distance. As pointed out in previous discussion, similarities between individuals can be a useful indicator of how much users likes/trusts each other; hopefully, with similarity properties, we can predict link weights by using the vectors. Here we use inner product $p_u^\top q_v$ to estimate the link weight $W_{uv}$, and model their similarities using cosine distance.

### 3.2 Weight Prediction by Matrix Decomposition

Following this idea, we define how to find such vectors and how to use them to predict link weights in WSNs. For a WSN with $n$ nodes, let matrix $W \in \mathbb{R}^{n \times n}$ denote the weight matrix of a WSN, and $W_{uv}$ is the weight of link $u \to v$ in the network. Let

$$P = (p_1, p_2, \ldots, p_n)^\top \in \mathbb{R}^{n \times K}, \tag{1}$$

$$Q = (q_1, q_2, \ldots, q_n) \in \mathbb{R}^{K \times n}. \tag{2}$$

The goal is to find $P, Q$ s.t.

$$W = PQ, \tag{3}$$

By this definition, we are using the inner product of vectors as the indicator of link weights.

Furthermore, with real-world networks, some networks are generally more positive and individuals tend to give positive evaluations (i.e. positive link weights), and other might not be friendly. Similarly, some individuals may tend to give other more negative evaluations out of their "habit", besides the part someone may does deserve the negative evaluation.

With these factors taken into consideration, we introduce some bias terms when estimating weights. Let $\mu$ denote the average known link weight in the WSN, $i.e.$ the bias for the network. Let $b_u^{(f)}$, $b_v^{(g)}$ denote respectively the bias of individual $u$ evaluating others, and $v$ being evaluated, where $b^{(f)}, b^{(g)} \in \mathbb{R}^n$. Then the estimate of link weight is

$$E_{uv} = \mu + b_u^{(f)} + b_v^{(g)} + p_u^\top q_v. \tag{4}$$

After splitting the links into training and test set, we can formalize the problem as finding corresponding $(b^{(f)}, b^{(g)}, P, Q)$ that minimize the cost $J(b^{(f)}, b^{(g)}, P, Q)$, where

$$\begin{aligned} J(b^{(f)}, b^{(g)}, P, Q) = & \sum_{uv \in \text{training}} (W_{uv} - E_{uv})^2 \\ & + \lambda \sum_u (||b_u^{(f)}||^2 + ||b_u^{(g)}||^2 + ||p_u||^2 + ||q_u||^2). \end{aligned} \tag{5}$$

Note that regularization terms are added to address the overfitting problem.

The parameters are first initialized to zero, and updated based on gradient descent (GD) method. Here we use batch gradient descent, but methods like mini-batch GD can be applied to reduce the training time.

Repeat until convergence:

$$\begin{aligned} b^{(f)} &:= b^{(f)} - \alpha \frac{\partial J}{\partial b^{(f)}} & \quad b^{(g)} &:= b^{(g)} - \alpha \frac{\partial J}{\partial b^{(g)}} \\ P &:= P - \alpha \frac{\partial J}{\partial P} & \quad Q &:= Q - \alpha \frac{\partial J}{\partial Q} \end{aligned}$$

where $\alpha$ is the learning step, and the partial derivatives can be calculated based on the following equations:

$$\begin{cases} \dfrac{\partial J}{\partial b_i^{(f)}} = -\sum_{uv} 2I(u = i)(W_{uv} - E_{uv}) + 2\lambda b_i^{(f)} \\ \dfrac{\partial J}{\partial b_j^{(g)}} = -\sum_{uv} 2I(v = j)(W_{uv} - E_{uv}) + 2\lambda b_j^{(g)} \\ \dfrac{\partial J}{\partial p_i} = -\sum_{uv} 2I(u = i)(W_{uv} - E_{uv})q_v + 2\lambda p_i \\ \dfrac{\partial J}{\partial q_j} = -\sum_{uv} 2I(v = j)(W_{uv} - E_{uv})p_u + 2\lambda q_j \end{cases}$$

Here $I$ denotes the indicator function, i.e.

$$I(x) = \begin{cases} 1, x \text{ is true,} \\ 0, \text{ otherwise.} \end{cases}$$

### 3.3 Comparison with Fairness-Goodness

Recall that Fairness and Goodness method predicts the weights by calculating the fairness and goodness of each node in the network and estimating the weight for link $(u, v)$ as $w_{uv} = f(u)g(v)$, which can be viewed as a latent factor model with 1 latent factor, $i.e.$ $K = 1$. In this case, $P = F \in \mathbb{R}^{n \times 1}, Q = G \in \mathbb{R}^{n \times 1}$, and the latent factor model shares the same intuition as Fairness and Goodness model, where $P$ denotes the fairness of each node (how fair is a node in rating other nodes), and $Q$ represents the goodness of each node (how likely is the node to be trusted by others).

As mentioned previously, one of the problem with the Fairness and Goodness model is that it lacks the ability to correct the prediction even if it has the wrong prediction in the known links, $i.e.$ for known link $uv$ we have $f(u)g(v) \neq W_{uv}$. It is hard to make exact prediction on the link weight, but the model metrics can be improved by correcting the wrongly predicted signs for cases that $\text{sign}(f(u)g(v)) \neq \text{sign}(W_{uv})$ or minimizing $|f(u)g(v) - W_{uv}|$. Therefore, besides finding the fairness and goodness metrics $P$ and $Q$, our model allows making correction to the fairness and goodness scores with the weights of the known links. To do so, the parameters are first initialized with the result of Fairness-Goodness method (instead of initialing all parameters to 0), and gradient descent is performed as shown in Section 3.2 to minimize the difference between the true weight and the corresponding matrix entry.

In the more generalized case, we set $K = 3$, and use the inner product $p_u^\top q_v$ to indicate their evaluations, the directions of such vectors can be seen as the indicator of similarities. Say $n(p_u)$ is the orthogonal plane for $p_u$, and it divides the space into two parts. If $q_v$ is in the same part with $p_u$, the inner product is positive and they can be quite similar; if $q_v$ is in the other part, the inner product is negative and they can be two completely different kinds of individual. Thus, the $K$-dimensional space allows the freedom for nodes to have a bunch of neighbors.

## 4 Model Extensions

In this section we discuss two possible extensions for the model, sign correction and time factor. They both conform with intuition and the real impacts are examined and analyzed in the experiments.

### 4.1 Sign Correction

In the proposed model we directly characterize the link weights by inner product, but sometimes this can cause problems like incorrect sign, for example trust is predicted as distrust. Since weight prediction is a harder task than sign prediction, we would like to add some correction for sign prediction to the model, which can be achieved by an extra penalty term for predictions with incorrect signs.

The modified cost function is

$$\begin{aligned} J(b^{(f)}, b^{(g)}, P, Q) = & \sum_{uv \in \text{training}} (W_{uv} - E_{uv})^2 \cdot (\tau(W_{uv}, E_{uv}) + 1) \\ & + \lambda \sum_u (||b_u^{(f)}||^2 + ||b_u^{(g)}||^2 + ||p_u||^2 + ||q_u||^2), \end{aligned} \tag{6}$$

where

$$\tau(x, y) = \begin{cases} I(y \neq 0), x = 0, \\ I(xy \leq 0), \text{ otherwise} \end{cases}$$

is 1 if and only if the prediction has incorrect sign.

Similarly, we can adopt gradient descent to obtain the biases and matrices. In experiments we will demonstrate the impacts of this correction.

### 4.2 Time Factor

In the previous analysis of link weight prediction, no information about link creation time is taken into consideration. Here, we would like to test whether the time information plays a role in weight prediction. Two competing hypotheses are proposed:

**Hypothesis 1**: Older links are more influential than newly created ones. Old links exist longer in the network, and newly formed link may be affected by the previous links. For instance, if all previous links of $v$ have high link weight, then it is likely that when the new link $uv$ is formed, $u$ is influenced by the former link weights. Therefore, the longer a link $uv$ exists in the network, it has more influence on other link weights.

**Hypothesis 2**: More recent links provide more reliable information for prediction, because the information of older links might be obsolete and link $uv$ is more influenced by the recently formed in-links of $v$.

The time information is encoded in the goodness metric calculation by adding linear time decay term with parameter $a$ or exponential time decay term with parameter $\alpha$.

$$g(v) = \frac{1}{\sum_{(u,v)|u \in in(v)} t(u, v)} \sum_{u \in in(v)} f(u) \times t(u, v) \times W(u, v)$$

Under these hypotheses, $t(u, v)$ can be calculated as follows. For each node $v$, sort in-link $uv$ in ascending link creation time, the number of in-link is $l$. For the $i^{th}$ node $u_i$ in the sorted list,

$$t_{\text{linear}}(u_i, v) = 1 - \frac{(1-a)}{l} i$$

$$t_{\text{exp}}(u_i, v) = e^{-\alpha i}$$

In the experiment section, we will examine how these two hypotheses perform on real data, if added into the fairness-goodness model. If it has improvements on prediction, we can hopefully use time factor in the matrix decomposition model by modifying the cost function. Parameter $a$ represents the lower limit we allow on linear decay. For instance if $a = 0.5$, we will have $t = 1$ for the first link (oldest in-link of $v$), and $t = 0.5$ for the last link (most recent in-link of $v$), which means we still take into account of half of the latest link's influence.

## 5 Datasets

In this section we provide some description and analyses of real-world datasets.

| Dataset | Vertices | Links | Range of link weight | % Pos. Links |
|---|---|---|---|---|
| Bitcoin OTC | 5,881 | 35,592 | -10 to 10 | 90.0 |
| Bitcoin Alpha | 3,783 | 24,186 | -10 to 10 | 93.6 |
| Wikipedia RfA | 9,654 | 104,554 | -1 to 1 | 83.9 |

Table 1: Statistics of weighted signed network datasets used in this project

## 5.1 Datasets description

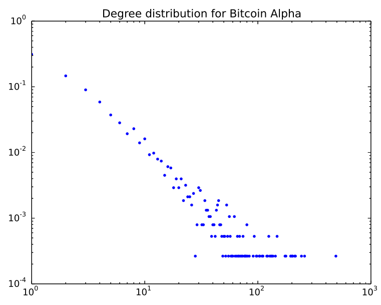
We select three real-world weighted signed network datasets to experiment with. The links in the networks are directed, where link $(u, v)$ means node $u$ judges node $v$, and positive and negative link weights have real meanings in networks. Statistics of the datasets are shown in table 1. Due to computation power limit, we will experiment with these three datasets, and we expect them to be representative because they are from different real world cases and have different properties.

**Bitcoin networks** Bitcoin is a decentralized digital currency that enables instant payment to anyone in the world. Due to the anonymity, there is a need to identify untrustworthy Bitcoin traders to avoid risky transactions. Bitcoin OTC[2] and Bitcoin Alpha[3] are two major Bitcoin rating datasets [7]. In both datasets, members in the Bitcoin community rate other members in a scale of $-10$ to 10 in steps of 1, where $-10$ denotes total distrust, and 10 means total trust. In the following analysis and experiments, the link weights are normalized to the range $-1$ to 1.

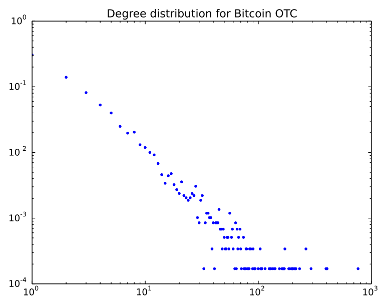
**Wikipedia network** Wikipedia editors submit request for adminship in order to be selected as administer of the website. Once the request is submitted, any Wikipedia member can cast a supporting, neutral, or opposing vote. The Wikipedia Request for Adminship (RfA) network [8] has been built by crawling the votes from 2003 to 2013. The weights in this directed signed network are in the range of $-1$ to 1, where $-1, 1$ correspond to negative and supporting votes respectively. The dataset is also available online[4].

## 5.2 Dataset analysis

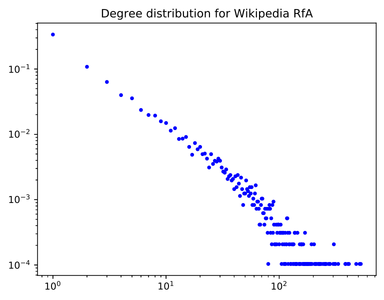
**Degree distribution** Figure 1 shows the degree distribution of these networks, where the x-axis is the degree in log scale, and the y-axis is the proportion of nodes in log scale. These networks all exhibit the power-law degree distribution.



(a) Bitcoin Alpha



(b) Bitcoin OTC



(c) Wikipedia RfA

Figure 1: Degree Distribution for Bitcoin Networks

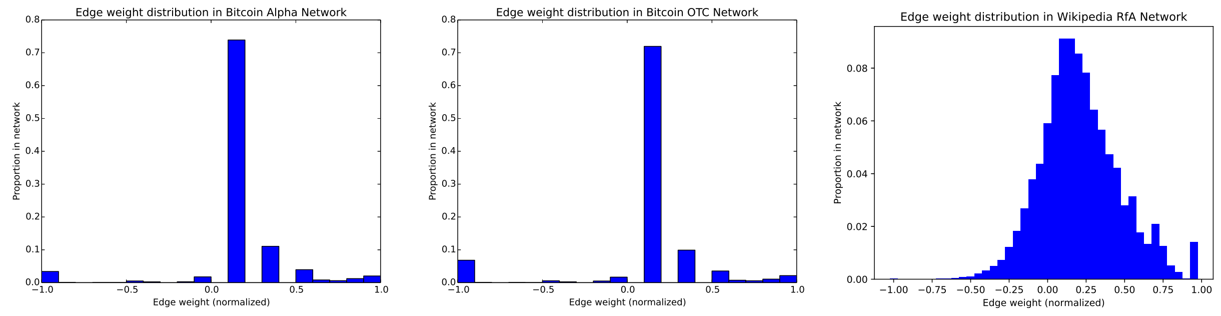
**Link weight distribution** Figure 2 shows the link weight distribution of these three networks (after normalization). After normalizing, the link weights in the two Bitcoin networks ranges from $-1$ to 1 in 0.1 step. The link weight in Wikipedia RfA network is more continuously distributed in the $-1$ to 1 range, and we draw the figure by counting the number of links in each bucket, where each bucket represents a weight range of 0.05.

Both Bitcoin networks share similar weight distribution, and the distribution is very uneven. A peak at 0.1 to 0.2 is observed, which means links with weights of 0.1 to 0.2 are more than 70% in both networks. A few links have weights $-1$ or $+1$; other weights are rare in the networks. In the Wikipedia RfA network, we also observe a peak around 0.1. Here most links have weights around 0.1 but not exactly 0.1. The weight distribution approximates a Gaussian distribution shape.

[2]http://snap.stanford.edu/data/soc-sign-bitcoinotc.html
[3]http://snap.stanford.edu/data/soc-sign-bitcoinalpha.html
[4]https://snap.stanford.edu/data/wiki-RfA.html

From Table 1, we can see that all three datasets have more than $80\%$ positive links. Based on the statistics, it is clear that most users give conservative positive scores, which may because they trust the other users based on past experience, but can't be absolutely sure about other users' future behavior. There are not many conservative negative scores (for example, $-0.2$) in the Bitcoin networks, which is because people can identify untrustworthy users based on just one fraudulent transaction. The percentage of conservative negative score is higher in Wikipedia RfA network than in Bitcoin networks, which is because editors who are bad at explaining certain topics might be good at other topics, so people are more tolerant in the Wikipedia community.

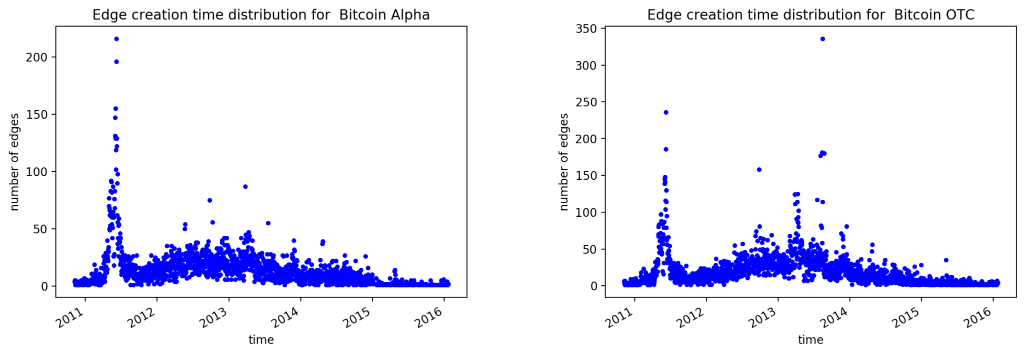
Since these two types of networks come from different real-world cases and are representative, we use them as the criteria to test the how the proposed model generalizes.



(a) Bitcoin Alpha (b) Bitcoin OTC (c) Wikipedia RfA

Figure 2: Link Weight Distribution for Bitcoin Networks

**Temporal information** With the two Bitcoin networks, temporal information of links is also provided. Figure 3 shows the creation time distribution of links. The time range spans from 2011 to 2016, and although with some peaks, most part it is flat, which means the network evolves gradually through the years. Since the Wikipedia network does not have the link creation time information, here we only explore on the Bitcoin networks.



(a) Bitcoin Alpha (b) Bitcoin OTC

Figure 3: Link Creation Distribution for Bitcoin Networks

## 6 Experiments

In this section, we explore the problem of predicting the weight of signed links in WSN. In the experiment analysis, we show that our matrix decomposition based algorithm **outperforms the state-of-the-art algorithm** based on the accuracy on three real world datasets.

### 6.1 Algorithms

**Reciprocal** We implemented the reciprocal algorithm as a baseline. This simple algorithm predicts the weight for link $(u, v)$ as the weight of reciprocal link $(v, u)$ if the reverse link exists. If there's no reciprocal link, the predicted weight is $0$.

**Fairness-Goodness** As described in related works, Fairness-Goodness model is the state-of-the-art algorithm for weight prediction in WSN. The main idea of the algorithm is to recursively define the fairness (how fair the node is in rating others) and goodness (how much the node is trusted) of each node in a graph. It has been demonstrated in previous work [7] that this algorithm has the best performance compared with other existing algorithms, including triadic balance, triadic status, EigenTrust [9], TidalTrust [10], PageRank [11], etc.

**Matrix Decomposition** As illustrated in the previous section, we propose to use matrix decomposition to predict link weights in WSN. We set $K = 3$, so the node properties are represented in $3-$dimensional vectors. In (batch) gradient descent, we set step size $\alpha = 0.01$, regularization parameter $\lambda = 0.1$, and the optimization runs 100 iterations. Initially the parameters are all 0. We also experiment with $K = 1$, where the vectors become scalars. In this case, we initialize the parameters with Fairness-Goodness parameters as a starting point of training. For better clarity of how much the similarities contribute to the model, we use both scalar version ($K = 1$) and vector version ($K = 3$) in following experiments.

## 6.2 Evaluation metrics

We use RMSE and PCC as evaluation metrics, to respectively depict how close the predicted weights are to ground truths, and how well the model maintains the weight relations (larger or smaller) between links.

**RMSE** Root mean square error (RMSE) is calculated between the ground truth link weight and the predicted link weight. When the weight of $n$ links are predicted, the reported RMSE is calculated as

$$\text{RMSE} = \frac{1}{n}\sum_{i=1}^{n}(W^{(i)}_{\text{truth}} - W^{(i)}_{\text{predict}})^2.$$

The performance of algorithm is better when the RMSE value is lower.

**PCC** Pearson's correlation coefficient (PCC) measures the linear association between two arrays. When we predict the weight of $n$ links, let $x$ be the array of length $n$ where each element is a ground truth value of link weight, and $y$ be the predicted array. Then the reported PCC is

$$\text{PCC} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y},$$

where cov(x,y) is the covariance, and $\sigma_x, \sigma_y$ are the standard deviations of each array. When PCC value is larger, it means that the trend of two arrays are more similar, thus the result is better.

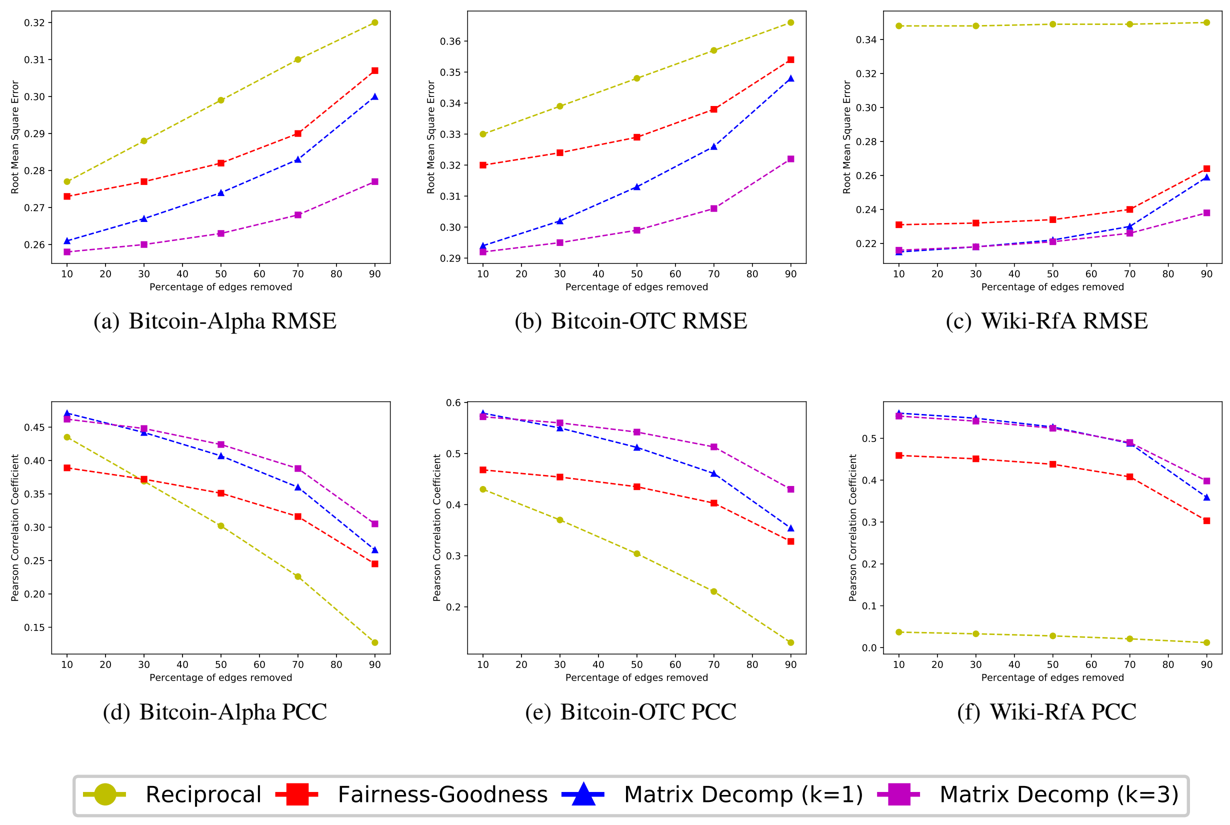
## 6.3 One-link Prediction

| Dataset | Method | RMSE | PCC |
|---|---|---|---|
| Bitcoin-Alpha | Reciprocal | 0.280 | 0.449 |
| | Fairness-Goodness | 0.291 | 0.385 |
| | Matrix Decomp (K=1) | 0.249 | 0.486 |
| | Matrix Decomp (K=3) | **0.236** | **0.498** |
| Bitcoin-OTC | Reciprocal | 0.348 | 0.418 |
| | Fairness-Goodness | 0.322 | 0.474 |
| | Matrix Decomp (K=1) | 0.280 | 0.614 |
| | Matrix Decomp (K=3) | **0.278** | **0.619** |
| Wiki-RfA | Reciprocal | 0.349 | 0.028 |
| | Fairness-Goodness | 0.236 | 0.441 |
| | Matrix Decomp (K=1) | **0.212** | 0.535 |
| | Matrix Decomp (K=3) | 0.218 | **0.578** |

Table 2: Prediction result comparison between algorithms

For one-link prediction, in each trial, we randomly remove one link from the network, and predict the weight of this link based on the rest of the network. We do 1000 trials of experiments on each network, and report the averaged RMSE and PCC of each network in Table 2. The best results among the different algorithms is in bold. From the table we can see that generally, reciprocal has the worst and the most unstable RMSE and PCC, and Fairness-Goodness model performs better and more stable than it; compared to these baselines, the proposed method has much lower RMSE and much higher PCC, and generally the version with $K = 3$ performs better than that with $K = 1$. These experiments demonstrate that the prediction accuracy of the proposed method is much better than existing models.

## 6.4 Multi-link Prediction



(a) Bitcoin-Alpha RMSE (b) Bitcoin-OTC RMSE (c) Wiki-RfA RMSE

(d) Bitcoin-Alpha PCC (e) Bitcoin-OTC PCC (f) Wiki-RfA PCC

Figure 4: This figure shows the performance of different algorithms when predicting different percentage of links in the graph. We observe that in all networks and in both metrics, matrix decomposition algorithm gives the best results.

For multi-link prediction, we randomly remove a fraction of links from the network, compute features based on the remaining links, and then predict the weights of removed links. We remove $10\%$ to $90\%$ links in steps of $20\%$ to explore the effect of missing links from the network and examine the robustness of models.

Figure 4 shows the results of our experiments. Recall that smaller RMSE and larger PCC demonstrate better prediction. Generally, Fairness-Goodness model provides better prediction than Reciprocal, and Matrix Decomposition models predict even better. Both Matrix Decomposition models consistently gives lower RMSE and higher PCC in all datasets for all percentage of missing links. Besides, comparison between these three scenarios shows that Matrix Decomposition is more robust when the known weights get fewer and fewer, and is more robust for different real-world cases. This can be useful for networks that are very sparse. Notice that the $K = 1$ version of proposed model performs slightly worse than the $K = 3$ version. This is because $3-$dimensional vector captures richer information about the feature of the nodes.

## 6.5 Sign Correction

In this part we examine the extension illustrated in Section 4.1. Unfortunately, on Bitcoin-Alpha dataset, some experiments of $K = 3$ version of proposed model show no real improvement (<0.001 in RMSE and PCC). To analyze the reason of this phenomenon, we take a typical run for Bitcoin-Alpha dataset, $50\%$ link prediction, and proposed model with $K = 3$. After extracting all the predictions with incorrect signs, we plot the distribution of their difference with true weights in Figure 5. The y-axis is the percentage in all predicted links.

Combining Figure 5 with Figure 2(a), we can easily deduct the situations of three peaks in the figure (because there are only three peaks in link weight distributions). As in shown in Table 3, we estimate the truth, prediction, and percentage for three peaks. Apparently, the sign correction can have only very small impacts on the first two situations, for the predicted weights and true weights are already very close (the penalty can be very small), and it can possibly improve the prediction in the third situation. However, due to the rarity of the case, and the fact that this penalty disappears once the sign is correct, the improvement can be too small to present in results.

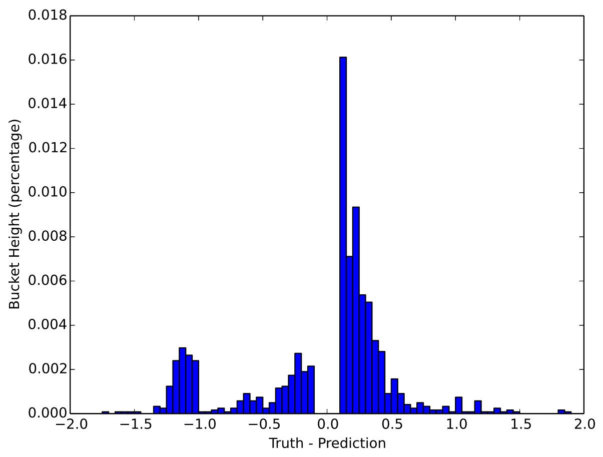


Figure 5: Weight difference between truth and prediction, for predictions with incorrect signs

In fact, assume that we correct the sign of all links in the these situation, the improvement of RMSE is loosely bounded by

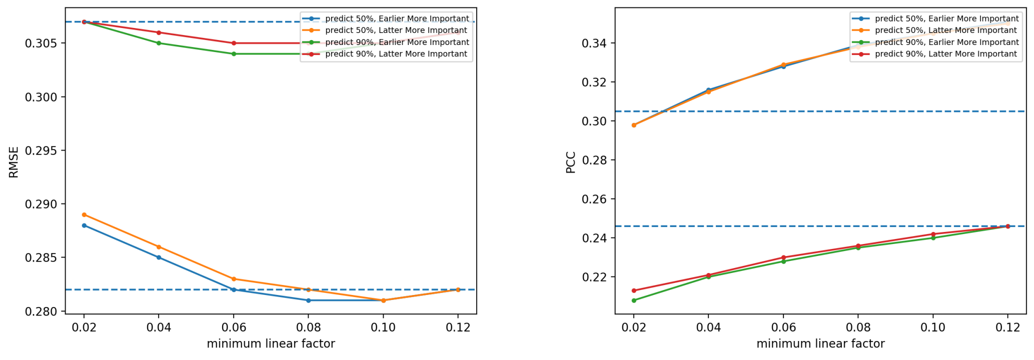
$$5\% \times 0.1^2 + 2\% \times 0.2^2 + 1\% \times 0.2^2 = 0.0017.$$

Due to the complexity of cost function, we expect real improvements to be much less than this bound, and therefore possible improvement is too little compared to differences between models. (Notice that here we assume the corrected weight is very much near 0, because the sign correction term disappears once the sign is correct, in which case it has the tendency of predicting incorrect-sign-weights for these links just as before).

Generalizing from above analysis, the sign correction term upon the proposed model has very limited impact of improving prediction results.

| Truth - Prediction | Truth | Prediction | Percentage |
|---|---|---|---|
| 0.1∼0.2 | +0.1 | -0.1∼0 | ∼5% |
| -0.3∼-0.1 | -0.1 | 0∼0.2 | ∼2% |
| -1.2∼-1.0 | -1.0 | 0∼0.2 | ∼1% |

Table 3: Situations for three peaks in Figure 5

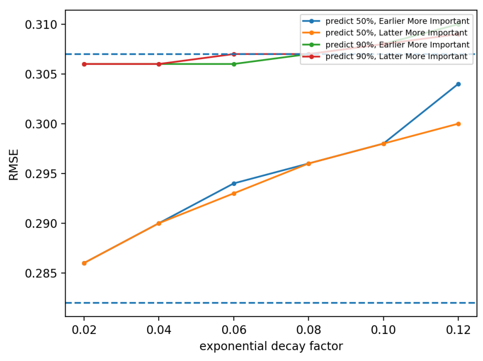


(a) RMSE (b) PCC

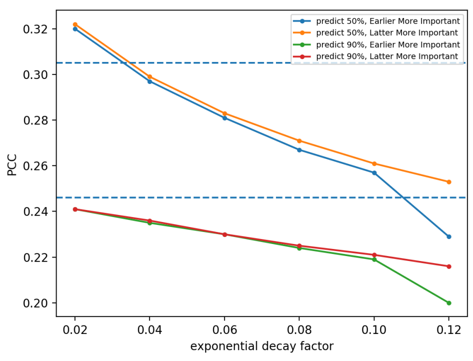
Figure 6: Linear Time Decay

## 6.6 Time Factor

Here we experimented on the time factor based on the two competing hypothesis and perform both linear decay and exponential decay on the goodness calculation. Figure 6 and Figure 7 show the RMSE and PCC when predicting $50\%$ and $90\%$ of the links,



(a) RMSE



(b) PCC

Figure 7: Exponential Time Decay

and the horizontal line shows the baseline of Fairness-Goodness model with no time information provided. It can be seen from the data that the two competing hypotheses show similar result, and neither has clear improvement compared with the baseline.

One possible explanation is that the time information does not play an essential role in link weight prediction, since no clear difference is observed when putting high weights on the earlier links and assigning higher weights on the latter links. Moreover, from the degree distribution of the network it can be seen that the majority of nodes have very low in-degree, just having 1 in-link. In this case, the time decay term will not play the expected role in the calculation of goodness, since $t(u, v) = 1$. Therefore, the influence of time factor still remains to be explored.

## 7 Conclusion

In this project, we proposed to apply matrix decomposition model in WSN link weight prediction for the first time, and further explored possibilities to incorporate sign correction and temporal information into modeling. Through experiments on three real world datasets, we have demonstrated the effectiveness of our model with higher precision than the state-of-the-art algorithms. Moreover, we tested the accuracies of algorithms by varying the size of network and predicting $1$ link as well as $10\%$ to $90\%$ of the links, which proved the robustness of our algorithm against node sparsity in WSNs.

## References

[1] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370. ACM, 2010.

[2] Eric Gilbert. Predicting tie strength in a new medium. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1047–1056. ACM, 2012.

[3] Rongjing Xiang, Jennifer Neville, and Monica Rogati. Modeling relationship strength in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 981–990. ACM, 2010.

[4] Eric Gilbert. Predicting tie strength in a new medium. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1047–1056. ACM, 2012.

[5] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 211–220. ACM, 2009.

[6] Stavros Sintos and Panayiotis Tsaparas. Using strong triadic closure to characterize ties in social networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1466–1475. ACM, 2014.

[7] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

[8] Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. Exploiting social network structure for person-to-person sentiment analysis. *arXiv preprint arXiv:1409.2450*, 2014.

[9] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

[10] Yarden Katz and Jennifer Golbeck. Social network-based trust in prioritized default logic. In *AAAI*, volume 6, pages 1345–1350, 2006.

[11] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.