Turkish Journal of Electrical Engineering and Computer Sciences

Volume 24 | Number 4 Article 56

1-1-2016

Application of a time delay neural network for predicting positive and negative links in social networks

SAGHAR BABAKHANBAK

KAVEH KAVOUSI

FARDAD FAROKHI

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 2825 – 2837
© TÜBİTAK
doi:10.3906/elk-1409-191

Application of a time delay neural network for predicting positive and negative links in social networks

Saghar BABAKHANBAK[1], **Kaveh KAVOUSI**[2,*], **Fardad FAROKHI**[3]
[1]Department of Computer and Information Technology Engineering, Islamic Azad University, Garmsar Branch, Garmsar, Iran
[2]Institute of Biochemistry and Biophysics, University of Tehran, Tehran, Iran
[3]Department of Electrical Engineering, Islamic Azad University, Central Tehran Branch, Tehran, Iran

Received: 30.09.2014 • **Accepted/Published Online:** 11.12.2014 • **Final Version:** 15.04.2016

Abstract: The structure of online social networks cannot in most cases be defined only by the relationships among its members. The interuser relations on the website of social networks are often a mixture of positive and friendly interactions, such as trust and interest, and at the same time negative interactions, such as mistrust and lack of interest. One of the issues with signed social networks is the prediction of edge signs. Some of the most recent studies have tried to extract features of the users and their relations with their neighbors to solve this problem. The results of such works demonstrate a relative success for such efforts. The present paper is an effort to offer a rather new approach to solve the problem that has not been addressed yet. The approach uses the distributed time delay neural network to present a model capable of predicting the sign of hidden or unknown edges. The goal is to have the neural network trained by available data and then assess its efficiency of the network. Our implementation on the actual datasets of Slashdot media shows that the algorithm, while simple, has a higher rate of precision in comparison to other existing methods.

Key words: Signed social networks, positive edges, negative edges, edge sign prediction, neural networks, distributed time delay neural network

1. Introduction

Many human networks are usually combinations of friendship and enmity relations, which are modeled in computer science by means of signed networks. Fortunately, the advent of social networks on the Web has provided people with a chance of expressing their trust or mistrust in other people's ideas, creating the notion of 'edge' in social networks. The goal of this paper is to predict the signs of edges in social networks. With increased social interaction in social networks there is one basic question of whom to trust. This has turned into a very critical challenge across the Web. Each user is faced with hundreds of opinions from other users, and many such ideas need a trust assessment. Trusted information may help users in making right decisions, ordering and filtering information, receiving recommendations, etc. We can use such information to offer suggestions to other users on who to trust.

Social interactions on the Web include both positive and negative connections. People form links with others to indicate support or endorsement, but they also link to signify disapproval or to express disagreement or distrust of the comments others make. We call such networks with both positive and negative links signed social networks [1]. While many interactions between positive and negative relationships in social networks are

*Correspondence: kkavousi@ut.ac.ir

clearly important, the vast majority of studies have considered only positive relationships, which means relations between people including friends and fans, followers, and colleagues. However, in many cases, it is important to also consider negative relationships, especially when studying interactions in social media: for example, discussion lists, lists of controversy, and disagreements on social sites are full of comments and therefore social sites are a place for conflict along with friendship. Generally, a social network contains both positive and negative connections, which are involved in a single structure [2,3].

New research has recently begun to examine the role of negative relationships in addition to positive relationships. For example, Wikipedia users can vote for or against a candidate for management [4], users on Epinions can express trust or distrust of others [5,6], and Slashdot participants can identify others to be either friends or enemies [2,7–9].

For a given link in a social network, we will define its sign to be positive or negative depending on whether it expresses a positive or negative attitude toward the receiver of the edge. The basic issue is as follows [2]: how will the sign of a given link affect link signs in its local vicinity, or more broadly in the whole network? In addition, what are the possible configurations of link signs in real social networks?

Answering these questions, we can understand the reason for using negative relationships in social networks. Indeed, one of the useful ways of growing social networks is proposing that users create new connections with other members of the network. This is based on mutual friends, common interests, and other shared properties. Social websites try to introduce a person to others that are familiar, reliable, and popular. There is an important challenge in this context: users may have positive or negative thoughts and tendencies towards others that have already formed and these relations are hidden. Related to this issue, it is important to estimate these tendencies from existing evidence in the network before offering any suggestions about other users [2,10].

In studying the edge sign prediction problem, we are following an experimental framework articulated by Leskovec et al. [2]. We approach the problem using a neural network technique. Using this neural network, we obtain significantly improved performance for the task itself.

The rest of this paper is organized as follows: Section 2 surveys the related works. Our problem and dataset are described in Section 3. In Section 4 our method is introduced. In Section 5 our implementation and results are described. Section 6 concludes our work.

2. Related works

As a part of the research on large and complex networks and their features, recently structural analysis of social networks has gained attention. In networks the nodes represent people and the edges represent communications between people, partnerships, and so on. A lot of studies on trust and friendship prediction have been done based on websites that allow users to show opinions about other users' context and comments, such as Epinions, eBay, Wikipedia, Essembly, and Slashdot.

An atomic propagation model was used in 2004 by Guha et al. [5] to predict trust and distrust between 2 users present in signed networks. Their model was the first major work on the edge sign prediction problem. They tested their model on an Epinions dataset. Leskovec et al. [2] reviewed the work done by Guha et al. They used a logistic regression classifier for edge sign prediction. It continued in their recent work [10] on the study of signed networks. They used famous psychology theories of structural balance and status. They presented a new assessment of these theories with a large dataset. Dubois et al. [11] used an inference algorithm that relies on a probabilistic interpretation of trust based on random graphs with a modified spring-embedding algorithm.

Kunegis et al. [8] studied the friend and foe relationships on Slashdot and computed global network properties, but did not evaluate theories of balance and status. Wang and Bulatov [12] used a machine learning approach to build a prediction model based on local features. They used peer opinions from trusted peers, which are personalized features. Liu et al. [13] extracted features from users' self-statuses and their relationships with neighbors. They used a binary classifier model by linear support vector machine. Massa et al. [6] used the Mole Trust metric, which reduces the prediction error for controversial users, to predict trust between users in Epinions. Burke et al. [4] presented a model of the behavior of candidates for promotion to administrator status in Wikipedia. Brzozowski et al. [7] studied user behavior from Essembly. They used decision trees to predict how a user will vote under the given conditions. Chiang et al. [14] used a supervised machine learning-based link prediction method that uses features derived from longer cycles in the network. Malekzadeh et al. [15] presented a game theoretic approach that models the formation of signed networks that contain both friendly and antagonistic relations. They proved the NP-hardness of computing the best response and gave an approximation for it by using quadratic programming and rounding its solution. Jihang et al. [1] stated that knowing the edge signs will provide us with a better understanding of user opinions in a social network. It is challenging due to the inadequate edge signs in the target network and the prohibitive cost of manual labeling. They adopted the transfer learning approach by leveraging a source network with abundant edge sign information but possibly under a different yet related distribution.

3. Predicting edge sign

The goal of this paper is to predict the edge signs following an approach used by Leskovec et al. [2]. In the dataset of any given social network the value of each edge is either positive or negative, therefore enabling us to classify them into 2 categories. Thus, the problem of predicting edge signs changes into a problem of classifying values into categories. We extracted a set of features from user statuses and their relationships with neighbors from the dataset for model training purposes and we then used a distributed time delay neural network to do the classification.

3.1. Dataset description

A signed dataset of the Slashdot social network collected by Leskovec et al. [2] was used for this study. The information is available in the set of large datasets on the Stanford website (http://snap.stanford.edu/data). The statistics for this dataset are presented in Table 1. All edges of this dataset are signed positive or negative. Slashdot is a news website related to technology on which users may label each other as friends or enemies. On this website the users rate each other with trust or mistrust signs that we take as positive or negative edges. A friendship relationship means that a user likes the opinions of another user, while an enemy relationship means that he is not interested in the opinions of the other user.

Table 1. Dataset statistics.

Slashdot dataset	
Nodes	82,144
Edges	549,202
+ Edges	77.40%
– Edges	22.60%

In this dataset distribution of signs of edges is imbalanced. As shown in Table 1, over 77% of edges are positive. This distribution of data may cause larger prediction error, which classifies edges more likely to the category with more samples. This matter does not influence our method much and results in Section 5 show this.

3.2. Features

In this study, we used the same features as used in [2,10]. These features are divided into 2 categories. The first category is based on degrees of nodes, which represent local relations of a node to the rest of the nodes. The second category is based on social psychology where we can understand the relationship between individuals a and b in terms of their relationship with another node, such as c [2,10].

In the first category of features, as we want to predict the sign of the edge from a to b, we consider outgoing edges from $a$and incoming edges to b. In this category there are 7 features. We show the number of incoming positive and negative edges to b with $d_{in}^{+}(b)$ and $d_{in}^{-}(b)$. Similarly, we use $d_{out}^{+}(a)$ and $d_{out}^{-}(a)$ to denote the number of outgoing positive and negative edges from$a$. We use $cn(ab)$ to denote the total number of common neighbors of a and b in an undirected sense, which means the number of nodes c such that c is linked by an edge in either direction with both$a$ and b. We will also refer to this quantity as the embeddedness of the edge (ab). Our 7 degree features are thus $d_{in}^{+}(b), d_{in}^{-}(b)\, d_{out}^{+}(a), d_{out}^{-}(a)$, and $cn(ab)$ with the total out degree of a and the total in degree of b, which are $d_{out}^{+}(a) + d_{out}^{-}(a)$and $d_{in}^{+}(b) + d_{in}^{-}(b)$ [2].

For the second category of features we consider each triad with the edge (ab) that consists one node c such that c has an edge either to or from a and also an edge either to or from b. The edge between c and a can thus be in either direction and of either sign, and the edge between c and b can also be in either direction and of either sign; this leads to 16 triads. Each of these 16 triad types may provide different evidence about the sign of the edge from a to b. We encode this information in a 16-dimensional vector specifying the number of triads of each type such that (ab) is involved as in [10]. In Figure 1 these 16 triad types are shown.

4. Learning methodology

In this paper an artificial neural network technique has been used to solve the problem of predicting edge signs. A serious limitation of most neural structures is their incapacity in facing the dynamic nature of data input in a network. For instance, the multilayer perceptron (MLP) neural network tackles nonlinear records in a cool manner, but it is a static network. In other words, the outputs of the network at each moment only depend on its inputs at the same moment, while in many practical conditions there is a need for a neural network that responds to a specific trail of patterns. For the same reason, in this study, a recurrent neural network algorithm is used. This neural network is the distributed time delay neural network (DTDNN). This network has a structure similar to that of a MLP network with the difference that there is one delayed connection in its input. The MLP is a feedforward neural network, and though the DTDNN is a feedforward network as well, there is a strip of delay in the inputs of each of its layers. In this network, we have used the K-nearest neighbor (KNN) algorithm and MLP neural network [16] for prediction and have compared the results of these 3 methods.

4.1. Distributed time delay neural network

A neuron and one or several layers of TDL and a nonlinear activation function and a multilayer feedforward network consisting of such units are together called a time delay neural network (TDNN). The first use of a

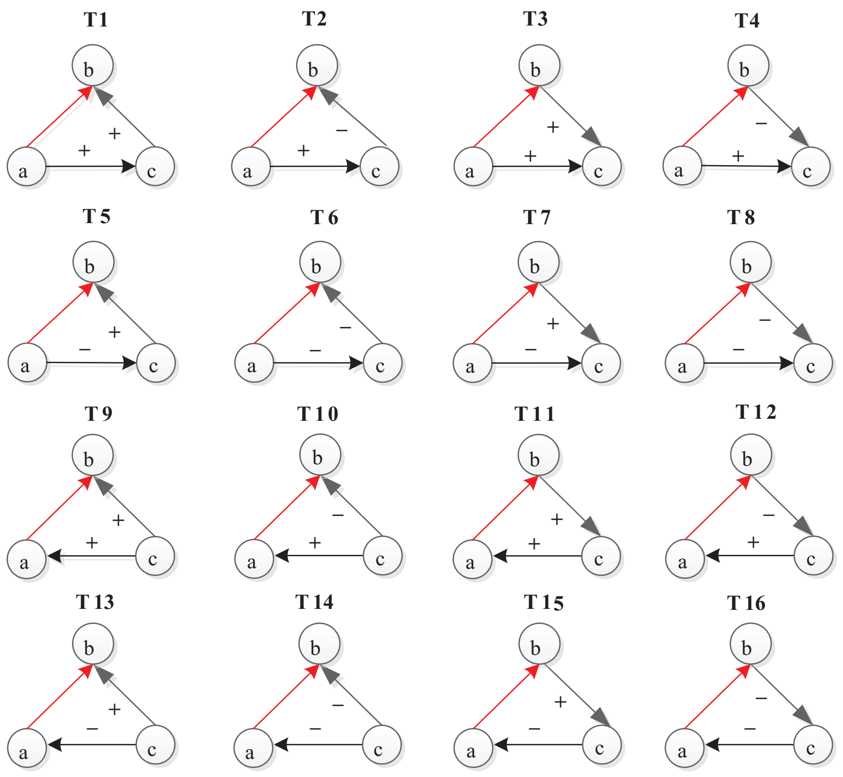

Figure 1. Sixteen triad types of features that are considered as a 16-dimensional features vector [10].

TDNN was in a paper aimed at phoneme detection written by Waibel et al. in 1988 [17]. This architecture was originally designed for processing speech sequence pattern in time series with local time shifts. Recently the TDNN has been used in various fields for identifying and classifying purposes. The main structural block for this type of network is a unit whose inputs have time delays [18]. Figure 2 provides a simple structure of delayed inputs into a feedforward network.

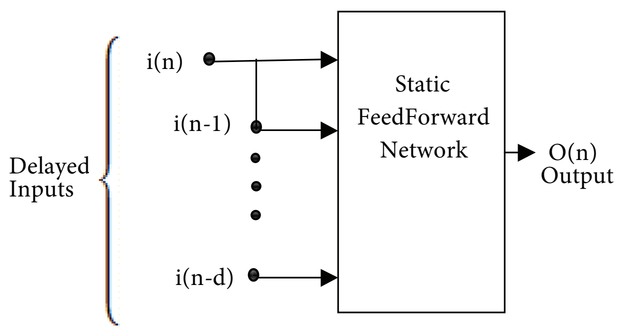

Figure 2. Delayed time inputs to a feedforward network.

In TDNN a basic unit that computes the weighted sum of its inputs and then passes this sum through a nonlinear function is modified by introducing delays d_1 through d_n as Figure 3 shows. The i inputs of this unit now will be multiplied by several weights; one is for each delay and another is for the undelayed input [17].

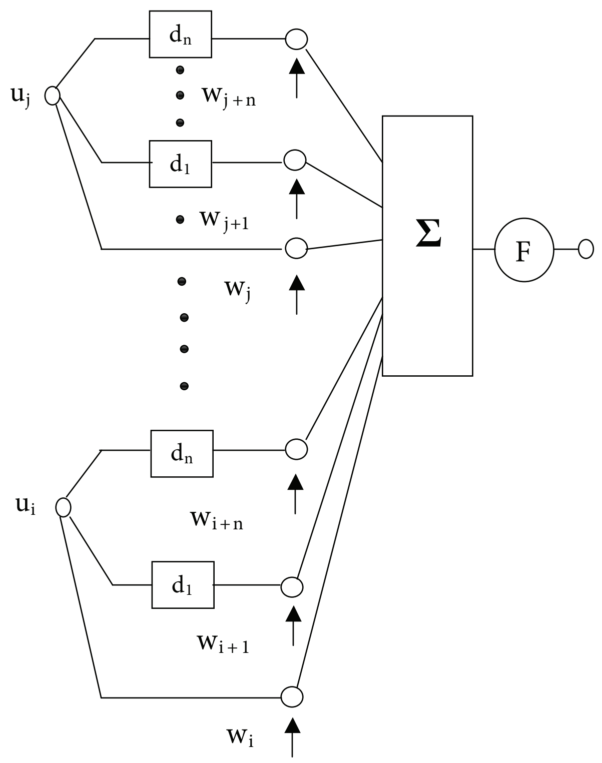

Figure 3. A time delay neural network unit [17].

In the neural networks, there are many techniques to train the network. In this neural network as a default the Levenberg–Marquardt (LM) [19] training algorithm is used to update the weights of the artificial neural network, which is a commonly used algorithm due to the fact that it can train networks rapidly and minimize error levels. The LM algorithm interpolates between the Gauss–Newton algorithm and the method of gradient descent. This algorithm outperforms simple gradient descent and other conjugate gradient methods in a wide variety of problems. This algorithm has been designed specifically to increase the learning speed of networks and is based on a Hessian matrix. The LM algorithm is a standard technique for numerical optimization, trying to reduce the calculations using the Hessian matrix. In this algorithm, the Hessian matrix is estimated to be as follows [20]:

$$H = J^T \times J$$

where J is the Jacobean matrix including the first derivatives of network errors in proportion to weights and biases. The Jacobian matrix for single neuron can be written as follows:

$$J = \begin{bmatrix} \frac{\partial \varepsilon_1}{\partial w_1} \cdots & \frac{\partial \varepsilon_1}{\partial w_n} & \frac{\partial \varepsilon_1}{\partial w_0} \\ \vdots & \vdots & \vdots \\ \frac{\partial \varepsilon_p}{\partial w_1} \cdots & \frac{\partial \varepsilon_p}{\partial w_n} & \frac{\partial \varepsilon_p}{\partial w_0} \end{bmatrix} \tag{1}$$

where p is the number of training patterns, $w = (w_1, w_2 \ldots w_n)$ is a vector of the weights, w_0 is bias of a neuron, and ε is an error vector (which means the difference between the actual and the required value of the network output for the individual pattern). The new configuration of weights in step $k+1$ is calculated as follows:

$$w_{k+1} = w_k - \left(J^T \times J + \lambda \times I\right)^{-1} \times J^T \times \varepsilon_k \tag{2}$$

Parameter λ is modified based on the development of error function ε. If the step causes a reduction in ε, we accept it. Otherwise, we change parameter λ, reset the original value, and recalculate w_{k+1} [19].

In this equation, w is the neural network weights; J is the Jacobean matrix, which must be minimized; λ is the number of training processes; and ε is the residual error vector.

In this network, the training and changing of weights continue until the mean squared error (MSE) is minimized or the efficiency curve slope reaches zero, or until the number of iterations defined for its training is completed. Thus, the criterion for MSE is defined by the following expression:

$$MES = \sum_{t=1}^{M} \sum_{s=1}^{N} (O_{st} - D_{st})^2 \tag{3}$$

In this equation O_{st} is the output of the network of the s^{th} neuron and t^{th} iteration, D_{st} is the desired output of the s^{th} neuron and t^{th} iteration, N is the number of neurons in output layer, and M is the number of test examples.

By default in a DTDNN, the hidden layer transfer function is a hyperbolic tangent sigmoid function that has this form:

$$Y_j = \frac{2}{(1 + \exp(-2 \times Z_j)) - 1} \tag{4}$$

Z_j is the sum of the weighted inputs to each neuron in the j^{th} layer with this form:

$$Z_j = \sum_{i=1}^{m} w_{ij} \times Y_i + b_j \tag{5}$$

where m is the number of output layer neurons, w_{ij} is the weight between layer i and j, Y_i is the output of the i^{th} neuron, and b_j is the bias value of neuron in the j^{th} layer. In this network, the transfer function of the output layer is a linear function.

4.2. K-nearest neighbor algorithm

In this paper, the KNN algorithm described in [21] is used. In [21] a learning method to optimize the parameters of the KNN algorithm, a classification method based on the Dempster–Shafer theory of belief functions, was proposed.

Dempster–Shafer theory of evidence

The mathematical theory of evidence, as a generalization of the Bayesian theory of subjective probability, provides a powerful representation of uncertainty and incomplete knowledge. The ability of this theory for uncertainty handling has led to many practical applications in pattern classification [22].

To understand the implications of this theory, suppose an arbitrary question is considered and the set of all possible answers to this question is shown with ϕ. In this case, all its subsets will be $P(\phi) = 2^{\phi}$. Any element of the power set of ϕ can be considered as a composite event of the frame of discernment. In Dempster–Shafer theory a body of evidence as a source of information is associated to each discernment $m(.) : P(\phi) \to [0,1]$ such that:

$$m(\emptyset) = 0 \tag{6}$$

$$\sum_{A\in P(\phi)} m\left(A\right) = 1 \tag{7}$$

More comprehensive information about this theory can be found in [21]. Now supposing that there is evidence from several sources, the Dempster rule of combination has been proposed by Shafer to combine 2 bodies of evidence $B1$and $B2$over the same frame of discernment ϕ. The basic assumption of the combination is independent sources. Suppose that $m_1(.)$ and $m_2(.)$ are the corresponding amount of masses of these bodies of evidence. So, $m\left(C\right)$ will be a combination of $m_1(.)$and $m_2(.)$ for any arbitrary set $\begin{cases} \neq\emptyset \\ \subseteq\phi \end{cases}$ such that $m(\emptyset) = 0$[20]:

$$0m\left(C\right) = \left[m_1 \oplus m_2\right]\left(C\right) = \frac{\sum_{A\cap B=C} m_1\left(A\right) \times m_2(B)}{\sum_{A\cap B\neq\emptyset} m_1(A)\times m_2(B)} = \frac{\sum_{A\cap B=C} m_1\left(A\right) \times m_2(B)}{1-\sum_{A\cap B=\emptyset} m_1(A)\times m_2(B)} \tag{8}$$

The fusion operator $\oplus$ also has commutative and associative property.

$$(m1\oplus m2 = m2\oplus m1) \tag{9}$$

$$(\left[m1\oplus m2\right]\oplus m3 = m1\oplus\left[m2\oplus m3\right]) \tag{10}$$

5. Simulation and results

We considered the Slashdot social network. While the Slashdot dataset has hundreds of thousands of nodes, we randomly extract an induced subnetwork that contains 2000 nodes. As shown in Table 1, in this dataset the ratio of positive to negative edges is 3:1. To choose 2000 records for the training set, we took 1500 records from the positive and 500 records from negative edges. For the DTDNN network we used a hidden layer of 15 neurons. We stopped the training of the network when the error was minimized. Thus, for 7 features we took an iteration rate of 38 and tapped delay of 1:8 that brought the best result. For 16 features of the triangles we took an iteration rate of 30 and tapped delay 1:8. In Figure 4 it is shown that the curve of the 7-features vector has become convergent after 38 steps.

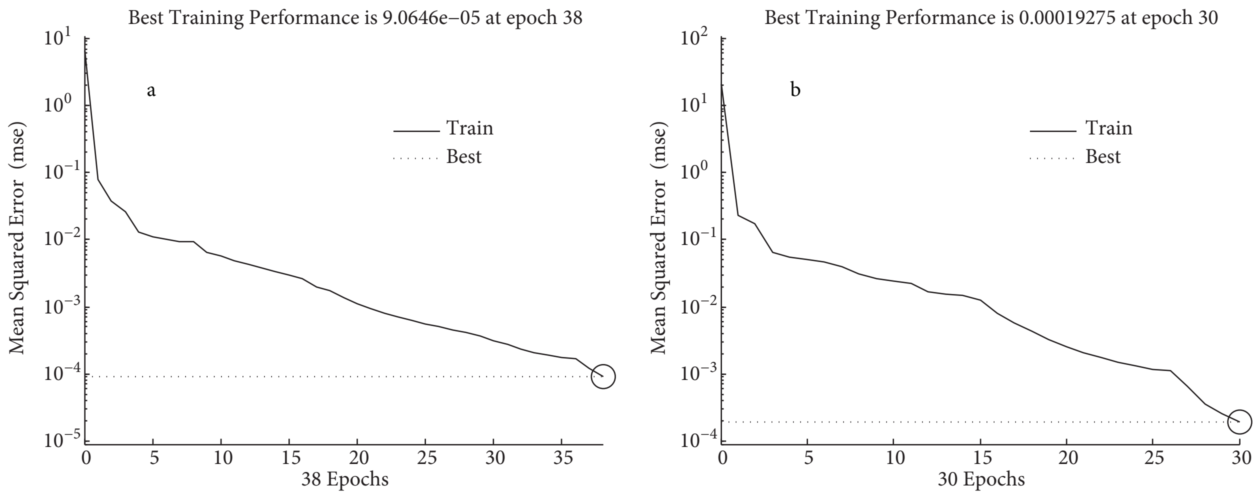

Figure 4. Convergence diagram in training process: (a) 7 features, (b) 16 features.

In Figure 5 in order to assess the validity of the neural network based model, linear regression of a model for 7 features and 16 triad features is shown and the results indicate that the regression coefficient of model after training, especially for 7 features, was near 1, which means that after learning the model showed better performance in comparison with previous work.

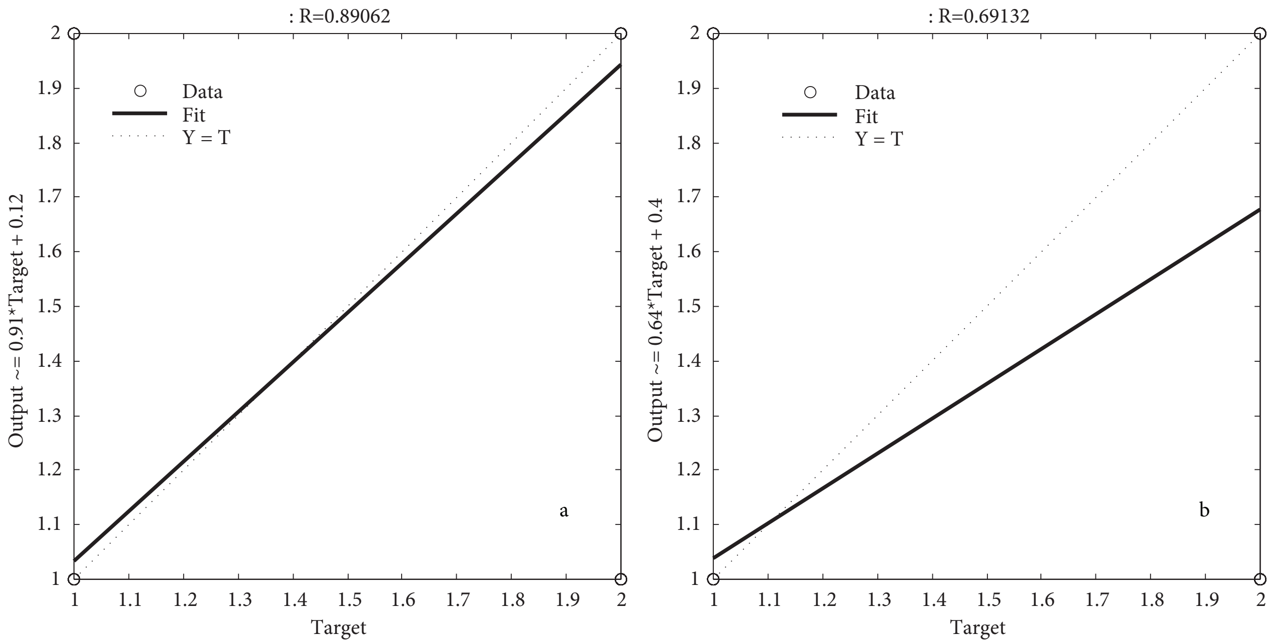

Figure 5. Network performance by linear regression: (a) 7 features, (b) 16 features. Network outputs are shown as empty circles. The best-fit line is shown with dots. Proportionality between the output and the target (desired output) is marked with a blue line and R is the correlation coefficient that shows correlation between the network response and the desired output.

The prediction performance could be assessed with different measures based on 4 basic parameters:

- TP (true positive): the number of positive edges predicted correctly.
- TN (true negative): the number of negative edges predicted correctly.
- FP (false positive): the number of negative edges predicted incorrectly as positive edges.
- FN (false negative): the number of positive edges predicted incorrectly as negative edges.

The sensitivity measure assigns the fraction of real positive edges correctly identified by the predictor. The specificity measure shows the fraction of negative edges that are correctly identified. The accuracy measure shows the proportion of true results (both true positives and true negatives) in the population and these are computed as follows:

$$Sensitivity = \frac{TP}{TP + FN} \times 100 \tag{11}$$

$$Specificity = \frac{TN}{TN + FP} \times 100 \tag{12}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{13}$$

Table 2 presents the results of this network. It is clear that in a good prediction both the sensitivity and specificity measures have to be of good precision. In Table 2 the sensitivity measure has a high level of precision, signifying that the network can classify positive edges with an accuracy of about 96%. According to these results it could be said that the model has completed the data training in a good manner. We also assess the accuracy of this method in each of the categories (positive and negative categories) and calculate the performance parameters. Table 3 shows the results for 7 features. Table 4 shows these results for 16 features. According to the results, the precision of the DTDNN method for the positive category is nearly 96%. This means that this method is suitable for predicting positive edges in this dataset. Similarly, the precision of predicting negative edges is nearly 94% for 7 features and 67% for 16 features. When predicting negative edges, this method does not perform as well as predicting positive edges, especially for 16 features. That is because the sensitivity rate of negative edges is 67.67%, while it is 96.18% for positive edges.

Table 2. The prediction results of distributed time delay neural network in 2 categories of features (7 features and 16 triad features).

DTDNN	Sensitivity	Specificity	Accuracy
7 features	96.55%	94.41%	96.06%
16 features	96.18%	67.67%	89.74%

Table 3. Prediction results of DTDNN in positive and negative categories with 7 features.

7 features	Sensitivity	Specificity	Accuracy
Positive category	96.55%	94.41%	96.54%
Negative category	94.41%	96.55%	94.41%

Table 4. Prediction results of DTDNN in in positive and negative categories with 16 features.

16 features	Sensitivity	Specificity	Accuracy
Positive category	96.18%	67.67%	96.17%
Negative category	67.67%	96.18%	67.67%

For the KNN algorithm we began from a $k=3$ neighborhood and calculated the accuracy of the predictions up to $k=14$. The optimal k for the 7-features vector was 14, for which the best result was obtained. For the 16-features vector the best result was obtained when $\phi\phi = 13$. As pointed out by Leskovec et al. [2], because triad features are only useful when the nodes on the 2 ends of the edge, i.e. a and b, have common neighbors, it is natural to expect the triad features to be more useful for edges with higher levels of embeddedness. Thus, other than testing the algorithm over the full dataset, we considered all subsets with various levels of embeddedness. We examined the efficiency of the algorithm in subsets with minimum levels of embeddedness of 0, 10, and 25. The classification accuracy is shown in Figure 6, where results are described in the 2 categories of features separately and for different levels of minimum embeddedness.

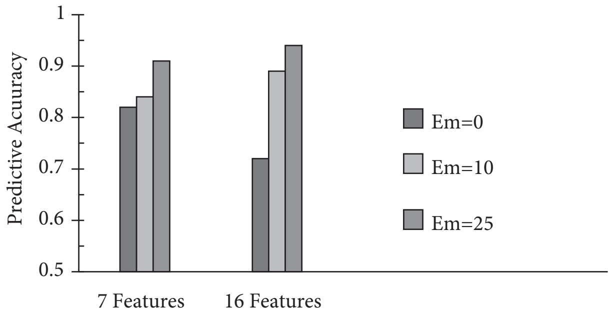

Figure 6. Accuracy of predicting the sign of the edge (ab) with given signs of all other edges in the network with k-nearest neighbor algorithm. Em is the embeddedness factor that has been explained before.

Several observations stand out. First, prediction based on the learned model significantly outperforms the results reported in [2] for the Slashdot dataset. The lowest error rate achieved in that paper was 13% for 7 features and 0.7% for 16 triad features, whereas we obtain error rates of 0.9% for 7 features and 0.6% for 16 triad features. It is perceivable that the triad features for edges with low embeddedness grades have smaller efficiencies than features of the degree, but for higher embeddedness values the triad features are far more efficient, and their efficiency increases with higher levels of embeddedness.

Second, as explained before, in a KNN algorithm the edge signs are estimated according to the signs of their closest neighbors. Neighbors are exactly the same features that have been used for training the model. That is why using this algorithm, in comparison with [2], and the MLP neural network and DTDNN algorithm, good results have been obtained. Prediction accuracy of these 3 methods applied on the full dataset is shown in Table 5.

Table 5. Prediction accuracy of 3 algorithms for 2 categories of features (with 7 and 16 triad features vector).

Algorithm	7 features	16 features
KNN	85.75%	80.49%
MLP	88%	74%
DTDNN	96.06%	89.74%

As Table 3 shows, the results of the MLP neural network in comparison with KNN and DTDNN, especially for 16 triad features, are weak. The main reason for this weakness is the static nature of this network. Among the 3 algorithms for 7 features and 16 triad features the best result is for DTDNN. Though we failed to produce good results with the KNN algorithm and the MLP neural network algorithm for the training dataset chosen by random selection, it must be taken into account that the failure is a sign that results of prediction may depend very much on the selection of the training and test dataset. The thing that has reduced the prediction accuracy is the method of selecting the training and test examples.

6. Conclusion

We studied the edge sign prediction problem in signed social networks that have both positive and negative links. We used the KNN algorithm and DTDNN to solve the problem of predicting edge signs. Our implementation of a signed dataset of the Slashdot website demonstrated that by using a DTDNN we successfully predicted signs for the 7-features vector with an accuracy rate of 96.06% and for the 16-features vector with an accuracy rate of 89.74%, which are the best accuracies in comparison to results of previous works. The results are particularly good because the neural networks used here have very simple implementation trends and calculations.

We know that the nature of the communications between nodes in social networks has a recursive property, and it cannot be said that communication between the nodes is only feedforward. However, in such networks it may be said that the relations of each node (one user, group, website, etc.) with other nodes are affected by the previous and next nodes. Because of these relations between nodes, using a recurrent neural network is much better than static neural networks like MLP etc.

The important point is that we got these results for the whole dataset. As we stated at the outset, Leskovec et al. [2] used the logarithmic regression classifier with these features for prediction, but we got better results. The best results of their work for the Slashdot dataset had 93% accuracy and they earned this result in the subsequent Slashdot dataset with different degrees of embeddedness, but we earned our results in the whole dataset. We did not limit our results to specific edges and select records of the training set with respect to the number of positive and negative edges, and tested our method on the rest of the records.

Our further work includes the following directions. We are interested in whether this method can be used in other kinds of social networks as well as other networks based on models and applications. One of the most important issues in the analysis of social networking websites is community extraction. As new research in the field of social networks, we want to study how networking communities can be extracted in social networks.

References

[1] Ye J, Cheng H, Zhu Z, Chen M. Predicting positive and negative links in signed social networks by transfer learning. In: Proceedings of the 22nd International Conference on World Wide Web; 13–17 May 2013. pp. 1477-1488.

[2] Leskovec J, Huttenlocher D, Kleinberg J. Predicting positive and negative links in online social networks. In: Proceedings of the 19th International Conference on World Wide Web; 26–30 April 2010. pp. 641-650.

[3] Newman MEJ. The structure and function of complex networks. SIAM Rev 2003; 45: 167-256.

[4] Burke M, Kraut R. Mopping up: modeling Wikipedia promotion decisions. In: Proceedings of the 2008 ACM conference on Computer Supported Cooperative Work; 8–12 November 2008. pp. 27-36.

[5] Guha R, Kumar R, Raghavan P, Tomkins A. Propagation of trust and distrust. In: Proceedings of the 13th International Conference on World Wide Web; 17–22 May 2004. pp. 403-412.

[6] Massa P, Avesani P. Controversial users demand local trust metrics: an experimental study on epinions.com community. In: Proceedings of the 20th National Conference on Artificial Intelligence; 2005. pp. 121-126.

[7] Brzozowski MJ, Hogg T, Szabo G. Friends and foes: ideological social networking. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 5–10 April 2008. pp. 817-820.

[8] Kunegis J, Lommatzsch A, Bauckhage C. The Slashdot Zoo: mining a social network with negative edges. In: Proceedings of the 18th International Conference on World Wide Web; 20–24 April 2009. pp. 741-750.

[9] Lampe CAC, Johnston E, Resnick P. Follow the reader: filtering comments on Slashdot. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2007. pp. 1253-1262.

[10] Leskovec J, Huttenlocher D, Kleinberg J. Signed networks in social media. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2010. pp. 1361-1370.

[11] Dubois T, Golbeck J, Srinivasan A. Predicting trust and distrust in social networks. Privacy, security, risk and trust (PASSAT). In: 2011 IEEE Third International Conference on Social Computing; 2011. pp. 418-424.

[12] Wang C, Bulatov AA. Inferring attitude in online social networks based on quadratic correlation. In: Yu JXU, Ravindran B, Pudi V, editors. Advances in Knowledge Discovery and Data Mining. Cambridge, MA, USA: AAAI Press, 2014. pp. 139-150.

[13] Liu F, Liu B, Wang X, Liu M, Wang B. Features for link prediction in social networks: a comprehensive study. In: IEEE International Conference on Systems, Man, and Cybernetics; 2012. pp. 1706-1711.

[14] Chiang KY, Natarajan N, Tewari A, Dhillon IS. Exploiting longer cycles for link prediction in signed networks. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management; 2011. pp. 1157-1162.

[15] Malekzadeh M, Fazli M, Khalilabadi PJ, Rabiee H, Safari M. Social balance and signed network formation games. In: Proceedings of KDD Workshop on Social Network Analysis; 2011.

[16] Haykin S. Neural Networks: A Comprehensive Foundation. New York, NY, USA: Pearson Education.

[17] Waibel A, Hanazawa T, Hinton G, Shikano K, Lang KJ. Phoneme recognition using time-delay neural networks. IEEE T Acoust Speech 1989; 37: 328-339.

[18] Reese MG. Application of a time-delay neural network to promoter annotation in the *Drosophila melanogaster* genome. Comput Chem 2001; 26: 51-56.

[19] More JJ The Levenberg-Marquardt algorithm: implementation and theory In: Watson GA, editor Numerical Analysis New York, NY, USA: Springer, 1978 pp 105-116.

[20] Dohnal IJ. Using of Levenberg-Marquardt method in identification by neural networks. In: Student EEICT; 2004. pp. 361-365.

[21] Zouhal LM, Denoeux T. An evidence-theoretic K-NN rule with parameter optimization. IEEE T Syst Man Cy C 1998; 28: 263-271.

[22] Kavousi K, Sadeghi M, Moshiri B, Araabi BN, Moosavi-Movahedi AA. Evidence theoretic protein fold classification based on the concept of hyperfold. Math Biosci 2012; 240: 148-160.