Node Classification in Social Network by Distributed Learning Automata

Ahmad Rahnamazadeh
Department of Electrical Engineering, Qazvin branch, Islamic Azad University, Qazvin, Iran
rahnamazade@gmail.com
Mohammad Reza Meybodi*
Department of Electrical Engineering, Amirkabir University, Tehran, Iran
mmeybodi@aut.ac.ir
Masoud Taheri Kadkhoda
Department of Electrical Engineering, Qazvin branch, Islamic Azad University, Qazvin, Iran
Taheri.masood@gmail.com

Received: 07/Sep/2016 Revised: 14/Nov/2016 Accepted: 05/Des/2016

Abstract

This paper presented a multiple Distributed Learning Automata (DLA) random walk model for node classification on a social network task. The purpose of this work is to improve the accuracy of node classification in social network by using of DLA. When dealing with large graphs, such as those that arise within the context of online social networks, a subset of nodes may be labeled. These labels can indicate demographic values, interest, beliefs or other characteristics of the nodes. A core problem is to use this information to extend the labeling so that all nodes are assigned a label.

Due to the high accuracy of local similarity measures, in the proposed algorithms, we will use them to build the transition matrix. As a standard in social network analysis, we also consider these networks as graphs in which the nodes are connected by edges and the transition matrix is used as weight value of edges. Now we partition this graph according to labeled nodes. Every sub-graph contains one labeled node along with the rest of unlabeled nodes. Then corresponding DLA on each partition. In each sub-graph we find the maximal spanning tree by using of DLA. Finally, we assign label by looking at rewards of learning automata. We have tested this algorithm on three real social network data sets. The result of Experiments show that the expected accuracy of a presented algorithm is achieved.

Keywords Social Network, Classification, Distributed Learning Automata, Node Labeling, Local Similarity Measure.

1. Introduction

The rise of online social networks in the past decade leads to generate more information to the people, ideas and their thoughts. Much of this information can be a model by labels that related with the people. These labels contain diverse information of nodes such as age, location, religion idea and etc.

Node classification in a social network in fact, assigns a label for unlabeled nodes from set of labels. There are many new applications for this kind of labeling, such as:

- Suggesting new connections or contacts to individuals, based on finding others of similar interests, demographics, or experiences.
- Recommendation system is to suggest objects (music, movies, and activities) based on the interests of other individuals with overlapping characteristics.
- Question answering systems which direct questions to those with the most relevant experience to a given question.
- Advertising systems is, which show advertisements to those individuals most likely to be interested and receptive to advertising on a particular topic.
- Sociological study is of communities, such as the extent to which communities form around particular interests or affiliations.
- Epidemiological study of how ideas and “memes” spread through communities over time.

Of course, these are just a few examples of the many different ways social network data is of interest to businesses, researchers, and operators of social networks. They have in common the aspect that knowing labels for individuals is a key element of each application.

So far, other methods are presented in two categories: iterative classification and random walk. The first category methods are based on iterative classification. Between researchers that have been done in this area, we can mention Neville and Jensen [1]. They originally used a Naive Bayesian classifier to derive labels in their instantiation of the ICA framework. An important special solution is the method of Macskassy and Provost[2], who used a simpler classification method based on taking a weighted average of the class probabilities in the neighborhood (effectively “voting” on the label to assign). Bhagat et al [3] proposed a method that considers the labeled nodes in the entire graph. This can be viewed as an instance of ICA using a nearest neighbor classifier to find a labeled node that is most similar to an unlabeled

* Corresponding Author

node being classified. Chakrabarti et al. in their method used features from neighboring documents to aid the classification, which can be viewed as an instance of ICA on a graph formed by documents [4].

The second category method is based on random walk. The node classification method of Zhu et al. [5] was proposed within the context of semi-supervised learning, where a symmetric weight matrix W is constructed using Eq. (1). More generally, we consider it to take as input graph $G(V,E,W)$, from which we derive the matrix $T = D^{-1}W$. Nodes V_l have initial labels Y_l from the label set Y.

$$w_{ij} = \exp(-\frac{\|v_i - v_j\|^2}{2\sigma^2}) \qquad (1)$$

Jaakkola and Szummer [6] considered the variation where the labeled nodes are not forced to be absorbing states. A recently proposed method by Zhou et al. [7] considers partitioning a graph based on both structural and attribute similarity of nodes on the graph. A central example is the work by Kleinberg and Tardos that describes the problem of Metric Labeling [8]. A different approach studied by McSherry is to use spectral methods (study of eigenvalues and eigenvectors) to recover a labeling [9]. Goldberg et al. make the observation that nodes may link to each other, even if they do not have similar labels [10]. Leskovec et al. [11] study the problem of classifying edges as positive and negative through the lens of two theories from social science literature: Balance and Status. Goyal et al. [12] studied a problem of edge labeling with applications such as viral marketing, where it is useful to know the influence that a user has on his neighbors. XiaohuaXu et al. [13] proposed algorithm based on multiple ant colonies for node classification. HuanXu et al. [14] presented a model by name of a factor graph. This model created a hidden graph model from main transition graph. In this model type of relation between nodes depend on direct or undirected relation in a transition graph. In continue they classified node with a loopy back propagation algorithms.

2. Problem Definition

Social network is defined by a graph $G = (V, E, W, Y)$, where V is set of nodes, E is set of edges that shown a relation between nodes in social network, W is weight matrix then shown similarity between nodes (more similarity, more weight) and Y is set of label of nodes. This label can be age, sex and other profile information and or other singed such as favorites.

In the classification in social network problem, we are given a graph $\mathrm{G(V,E,W)}$ with a subset of nodes $\mathrm{V_l \subset V}$ labeled, where V is the set of n nodes on the graph (possibly augmented with other features), and $\mathrm{V_u = V - V_l}$ is the set of unlabeled nodes. Here W is the weight matrix, and E is the set of edges. Let Y be the set of m possible labels, and $\mathrm{Y_l = \{y_1, y_2, \dots, y_l\}}$ be the initial labels on nodes in these V_l. The task is to infer label's Y on all nodes V of the graph.

Let $Y_l = \{y_1, y_2, \dots, y_l\}$ be the initial labels from the label set Y, on nodes in the set V_l.

For clarity, we provide some illustrative example of how (social) network data may be captured by a variety of choices of graph models:

Example: As an example of a different kind os a network, consider the picture and video sharing website, Flickr. Let graph $G(V,E,W)$ represent the Flickr user network, where:

- **Nodes V**: A node $v_i \in V$ represents a user.
- **Edges E**: An edge $(i,j) \in E$ between two nodes v_i, v_j could be an explicit link denoting subscription or friend relation; alternately, it could be a derived link where v_i, v_j are connected if the corresponding users have been co-viewed more than a certain number of picture or videos.
- **Node Labels Y**: The set of labels at a node may include the user's demographics (age, location, gender, occupation), interests (hobbies, movies, books, pictures), a list of recommended videos extracted from the site, and so on.
- **Edge Weights W**: The weight of an edge could indicate the strength of the similarity, could be created from local similarity measure, denoting subscription or friend relation.

3. Learning Automata and Distributed Learning Automata

3.1 Learning Automata

A learning automaton is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based upon a probability distribution kept over the action-set and at each instant, the given action is served as the input to the random environment. The environment responds to the taken action in turn with a reinforcement signal. The action probability vector is updated based upon the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized [15]. The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$ where a $\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents the finite set of inputs (actions), $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_r\}$ denotes the set of values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, \dots, c_r\}$ denotes the set of the penalty probabilities is called penalty set. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non-stationary environment. The environments depending upon the nature of the reinforcement signal b can be classified into P-model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values within the interval $[0,1]$ can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments, the reinforcement signal lies in the interval $[a,b]$. The relationship between the learning automaton and its random environment has been shown in Fig. (1).

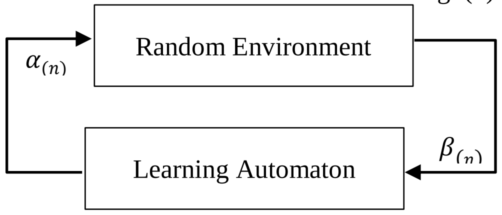


Fig. 1: The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families: fixed structure learning automata and variable structure learning automata. Variable structure learning automata is represented by a quadruple $\mathrm{LA} \equiv \{\alpha, \beta, p, T\}\}$ where $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_r\}$ is the set of inputs, $\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ is the set of actions, $T \equiv p(n+1) = T[\alpha(n), \beta(n), p(n)]$ is learning algorithm, and $p \equiv \{p_1, p_2, \dots, p_r\}$ is the Action probability vector. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ and $p(k)$ denote the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (2) and (3) is a linear learning algorithm by which the action probability vector p is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k. The action probabilities are updated as given in Eq. (2), when the chosen action is rewarded by the environment (i.e., $\beta(n) = 0$). When the taken action is penalized by the environment, the action probabilities are updated as defined in Eq. (3) (i.e., $\beta(n) = 1$).

$$p_i(n+1) = \begin{cases} p_i(n) + a.\left(1 - p_i(n)\right) & \alpha(n) = \alpha_i \\ p_j(n) - a.p_j(n) & \alpha(n) = \alpha_i \ \forall j \ j \neq i \end{cases} \tag{2}$$

$$p_i(n+1) = \begin{cases} (1-\mathrm{b}).p_i(n) & \alpha(n) = \alpha \\ \dfrac{b}{r-1} + (1-\mathrm{b}).p_j(n) & \alpha(n) = \alpha_i \ \forall j \ j \neq i \end{cases} \tag{3}$$

Where r the number of actions is can be chosen by the automaton; a and b denoted the reward and penalty parameters and determined the amount of increases and decreases of the action probabilities, respectively. If $a = b$, the recurrence Eq. (2) and Eq. (3) are called linear reward-penalty (L_{RP}) algorithm, if $a \gg b$ the given equations are called linear reward-e penalty ($L_{R\varepsilon P}$), and finally if $b = 0$ they are called linear reward-inaction (L_{RI}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

3.2 Distributed Learning Automata

A distributed learning automata (DLA) is a network of the learning automata which collectively cooperated to solve a particular problem. Formally, a DLA can be defined by a graph $\mathrm{DLA} = (\mathrm{V}, \mathrm{E})$, T and A_0 , where $\mathrm{V} = \{\mathrm{LA}_1, \mathrm{LA}_2, \dots, \mathrm{LA}_n\}$ is the set of learning automata, $\mathrm{E} \subset \mathrm{V} \times \mathrm{V}$ is the set of the edges in which edge e_{ij} corresponds to the action e_j of the automaton LA_i, T is the set of learning schemes with which the learning automata updated their action probability vectors, and A_1 is the root automaton of DLA from which the automation activation is started. An example of a DLA has been shown in Fig. (2). LA_j is activated when action j of LA_i is selected. The number of actions for LA_i equal with a number of outgoing edge from node LA_i.

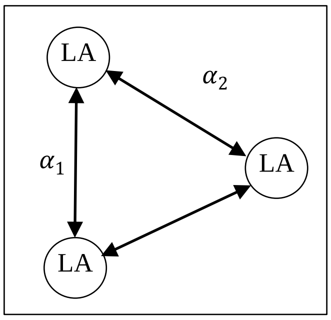


Fig. 2: Distributed learning automata

4. Distributed Learning Automates Random Walk

4.1 Similarity Matrix by Local Measure

Due to the good results of local similarity measures in various articles, in this paper we use, eight measures base on common friends or common relation to create a transition matrix. In continue we describe eight measures that used for this article and other articles.

Jaccard Coefficient: This coefficient one of the popular measures in a data recovery field [16, 17]. This coefficient defined as a ratio of a common friends in a union of a friend for two nodes. Equation of a coefficient shown in Eq. (4).

$$JC(x,y) = |\Gamma(x) \cap \Gamma(y)| / |\Gamma(x) \cup \Gamma(y)| \tag{4}$$

Where $\Gamma(x)$ is the number of a friend for node x.

Adamic/Adar Index: This measure is useful for find a relation between webpages, and is linked with a common features of two webpages [17, 18]. Equation of this index shown in Eq. (5).

$$AA(x,y) = \sum\nolimits_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log|\Gamma(z)|} \qquad (5)$$

Where Γ(z) is outgoing of a common friends between nodes x, y.

Hub promoted index: This measure is used for determine to overlap between two nodes [19]. This index defines as a ratio of a common friends to a minimum of friends between two nodes. Equation of this index shown in Eq. (6).

$$PI(x,y) = |\Gamma(x) \cap \Gamma(y)|/\min(|\Gamma(x)|, |\Gamma(y)|) \qquad (6)$$

Hub depressed index: This measure is the same before at with difference that the denominator is the maximum of friends between two nodes [20]. Equation of this index shown in Eq. (7).

$$HDI(x,y) = |\Gamma(x) \cap \Gamma(y)|/\max(|\Gamma(x)|, |\Gamma(y)|) \qquad (7)$$

Link weight: this measure containing two measures first sum of weight and second product of weight [20]. At the first for each node, edge weights were calculated separately according to Eq. (8). Then link weight calculated as summation or production of two node weight (Eq. (9) and Eq. (10)).

$$(x) = \frac{1}{\sqrt{1+\Gamma(x)}}, w(y) = \frac{1}{\sqrt{1+\Gamma(y)}} \qquad (8)$$

$$SW(x,y) = w(x) + w(y) \qquad (9)$$

$$PW(x,y) = w(x) \times w(y) \qquad (10)$$

Salton index: This measure defines as a ratio number of common friend to geometric mean of each node friends [17]. Equation of this index shown in Eq. (11).

$$SI(x,y) = |\Gamma(x) \cap \Gamma(y)|/\sqrt{|\Gamma(x)|.|\Gamma(y)|} \qquad (11)$$

Sorenson index: This measure define as ratio number of common friends to arithmetic mean of each node friends [17]. Equation of this index shown in Eq. (12).

$$SI(x,y) = |\Gamma(x) \cap \Gamma(y)|/\sqrt{|\Gamma(x)|.|\Gamma(y)|} \qquad (12)$$

In this study, we use measures defined above for create the transition matrix.

4.2 General Description of Proposed Algorithm

In this proposed algorithm, at first we use graph partitioning: each sub-graph containing labeled node and set of an unlabeled nodes. Now for each sub-graph is a corresponding network of distributed learning automata. Set of action for each automata is equally to be out-degrees of a node. In addition for better convergence, we let automates use the transition matrix value as the initial probability vector. Our object is achieved to be maximal spanning tree in each sub-graph**.**

In each sub-graph labeled node as root and random select any action from action set. This choice of action has two consequences:

1: activated automata on another side selected action and put the activated automata in an active group.

2: inactive selected action on another automata for avoid cycle in a graph

This trend continues to get all the automaton in the active group. Finally if cost of a tree improved, considers its environment as favorable response and all automata in a tree rewarded. Otherwise, all automata is fined. We use linear reinforcement learning in proposed algorithm. When the chosen action is rewarded by the environment, the action probabilistic update as given in Eq. (1) and when the chosen action is penalized by the environment, the action probabilistic update as given in Eq. (2).

Because of being inactive some action of automata for avoid of a cycle in a sub-graph, before selected one action, probabilistic vector divided to a summation possibility of active actions. This subject is shown in Eq. (13).

$$p_i = prob[\alpha(n) = \alpha_i | V(n),\ \alpha_i \in v(n)] = \frac{p_i(n)}{K(n)} \qquad (13)$$

Where $V(n)$ is set of active action's automata.

In the end of each iterative, we update the probabilistic action's vector because of activating, inactive action. This subject is shown in Eq. (14) and Eq. (15).

$$p_j(n+1) = p_j(n+1)K(n) \quad \text{for all j}, \alpha_j \in \mathrm{V(n)} \qquad (14)$$

$$p_j(n+1) = p_j(n) \quad \text{for all j}, \alpha_j \notin \mathrm{V(n)} \qquad (15)$$

The finishing condition of an algorithm achieved the product of weight selected tree edge to a threshold. In final unlabeled node given the label of winner automata (the most of acquired rewarded).

The general idea of our proposed algorithm is illustrated in Fig. (3), Fig. (4)

In the algorithm of Fig. (4) X_m is set of a labeled nodes and Y_m is set of labels. The target of algorithm is assigned labels to X_u from set of Y_m . P is a transition matrix calculated with LSM in section (4-1). For each sub-graph contains one labeled node as root and all other unlabeled nodes, finding MST by network of learning automata according to the algorithm Fig. (3).

```
Input: The node label $l$ and a sub-graph $G_l = (V,E,Q^t_l)$, where $V$ , $Q^t_l$ is the probabilistic rule at
generation t
Output: $V_{new}$ and $E_{new}$ , which describe a DLA;
$\Delta R$, wich is used for reward update
    1. $V_{new} \leftarrow \{v_l\}$, where $v_l$ is the labled node
       (root node) from $V$, $E_{new} \leftarrow \emptyset$.
    2. repeat
    3. Choose an edge $(v_j,v_k)$ randomly by
       action probability vector of automata
       (value in $Q^t_l$ probability) such that
       $v_j \in V_{new}$ and $v_k$ is not. (LA on $v_k$ is
       active)

    4. Remove all edge connected to $v_j$ (For
       avoid cycle in graph)

    5. Add $v_k$ to $V_{new}$ and  to $E_{new}$
```

Fig. 3: ALGORITHM: Finding Maximal Spanning Tree With DLA

```
Input: training set ($X_m$,$Y_m$), test set $X_u$
Output: $Y_u$
    1.  Initialize parameters;
    2.  Compute $P$ according to Similarty Indexes
    3.  Construct sub-graph G
    4.  While termination conditions is not met Do
    5.  For each node $v_l$ do
    6.  Crawled out recording path form root to unlabeled nodes (Algorithm Fig .(3))
    7.  Traverse this tree and update Reward $R(:,l)$ (if improved weight of tree)
    8.  Update $P$ Accroding to Learning Algorithm
    9.  End for
    10. End while
    11. Assign labels to $Y_u$ according to $R$
```

Fig. 4: ALGORITHM: Classifier of DLA Maximal Spanning Tree

To give a better explanation of the above algorithm, we show the detailed process based upon the example of Fig. (5).

V1 and V2 are a labeled nodes (White nodes) and V3, V4, V5 is an unlabeled nodes (Dark nodes). So we partition this graph to two sub-graph with root labeled node V1, V2 shown in Fig. (6), Fig. (7). Two DLA is assigned to sub-graphs DLAa, DLAb. Weight of each edge is calculated with Local Similarity Measure in section (4-1). Now in each sub-graph finding MST with labeled node as root vertex, by DLA and dividing reward and penalty to LA's in each DLA (Algorithm Fig. (3)). After initial generation of automata random walk, each unlabeled vertex will be attached by two kinds of rewards, and their intensities are given according to ΔR1: and ΔR2:. Fig. (7) Exactly illustrates the reward increments on the entire unlabeled graph after first generation. Both kinds of reward increment are computed according to their generated MSTs and the equation proposed. For example, in this sample in a vertex V3, kind of automata in DLAa is a dominant ($\Delta R_{13} > \Delta R_{23}$) then assign label V1 for V3 and likewise for the rest.

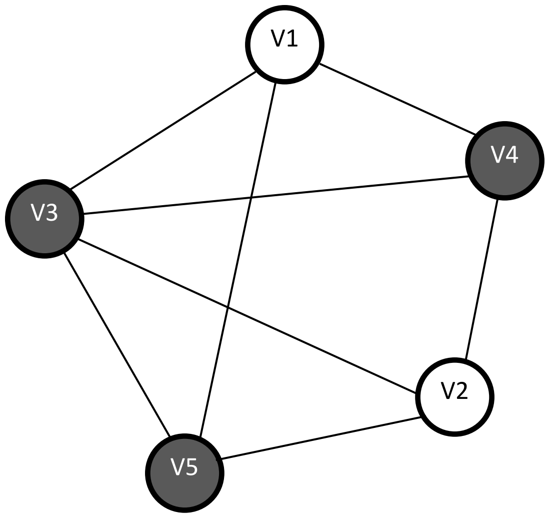


Fig. 5: A simple social network graph with 5 node and relation between

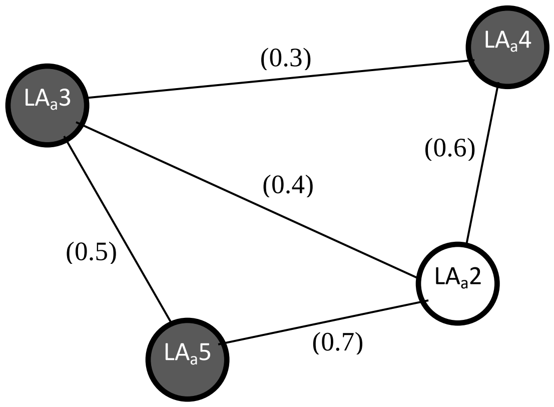


Fig. 6: Sub-graph with DLA_a and weight of relation by root V_2

4.3 Definition of Reward Matrix

In order to keep the rewards of belonging to each automaton, we use the matrix of rewards. Value of each column in our matrix is shown total rewards of each type learning automata. Value of each row in our matrix is shown total rewards of each labeled node for all types of learning automata. In another side equation in Eq. (16) shown reward of learning automate type l on unlabeled node I in tth iterative.

$$R_{il}^{t} = R^{t}(j,l) \tag{16}$$

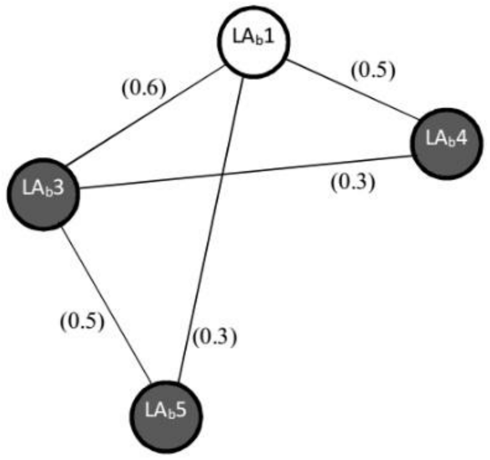
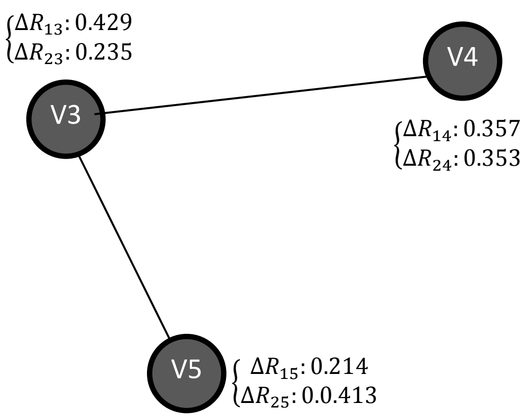


Fig. 7: Sub-graph with DLA_b and weight of relation by root V_1



$\begin{cases}\Delta R_{13}: 0.429 \\ \Delta R_{23}: 0.235\end{cases}$
V3
V4
$\begin{cases}\Delta R_{14}: 0.357 \\ \Delta R_{24}: 0.353\end{cases}$
V5
$\begin{cases}\Delta R_{15}: 0.214 \\ \Delta R_{25}: 0.0.413\end{cases}$

Fig. 8: Reward increment of each vertex after first generation of automata activation

4.4 Integrated Probabilistic Transition Rule

In the end of each iterative and activate every automata in per sub graph, if cost of a tree improved, considers its environment as favorable response and every automata in a tree rewarded. Otherwise, all automats is penalized. We use linear reinforcement a learning algorithm in proposed method.

5. Experiments

5.1 Test Datasets

The four data sets are obtained for experiment. First, data set publicly available from the SNAP data repository, Second and third data set getting from Network Economic group and related MySpace, Netlog social networks and fourth data set derived from Flickr social network by Elena Zheleva and Lise Getoor [21]. Detailed information is provided in Table (1).

Table 1: Used Dataset Detail

Dataset Name	***Number of Node***	***Number of Class***
Organization Network	46	7
Myspace	621	39
Netlog	1484	52
Flicker	14451	55

5.2 Test Results

We decided to node classification on four data set with various percentages of unlabeled node and local similarity measures. You can see the accuracy of probable label assigning in Table (2) to Table (5).

Table 2: Result of experiment by Organizational Network Dataset

		Percentage of initially known labels			
		50%	***67%***	***75%***	***80%***
Local similarity measure	Adamic/Adar	0.24	0.36	0.45	0.69
	Salton	0.27	0.47	0.45	0.67
	Sorenson	0.26	0.43	0.48	0.75
	Jaccard	0.34	0.42	0.55	0.79
	HPI	0.34	0.44	0.54	0.79
	HDI	0.31	0.40	0.56	0.76
	PW	0.37	0.40	0.52	0.85
	SW	0.39	0.47	0.58	0.92

Table 3: Result of experiment by Myspace Dataset

		Percentage of initially known labels			
		50%	***67%***	***75%***	***80%***
Local similarity measure	Adamic/Adar	0.25	0.35	0.44	0.73
	Salton	0.23	0.37	0.43	0.69
	Sorenson	0.28	0.40	0.45	0.73
	Jaccard	0.25	0.41	0.52	0.79
	HPI	0.31	0.42	0.58	0.81
	HDI	0.33	0.43	0.55	0.82
	PW	0.36	0.43	0.51	0.84
	SW	0.32	0.57	0.62	0.91

Table 4: Result of experiment by Netlog Dataset

		Percentage of initially known labels			
		50%	***67%***	***75%***	***80%***
Local similarity measure	Adamic/Adar	0.26	0.38	0.45	0.64
	Salton	0.24	0.40	0.41	0.62
	Sorenson	0.30	0.38	0.42	0.74
	Jaccard	0.29	0.39	0.44	0.73
	HPI	0.37	0.42	0.54	0.75
	HDI	0.26	0.43	0.49	0.78
	PW	0.33	0.43	0.64	0.88
	SW	0.32	0.56	0.64	0.9

Table 5: Result of experiment by Flickr Dataset

		Percentage of initially known labels			
		50%	***67%***	***75%***	***80%***
Local similarity measure	***Adamic/Adar***	0.25	0.36	0.40	0.61
	Salton	0.24	0.39	0.41	0.64
	Sorenson	0.31	0.38	0.44	0.74
	Jaccard	0.25	0.37	0.43	0.76
	HPI	0.28	0.41	0.58	0.76
	HDI	0.29	0.38	0.41	0.74
	PW	0.33	0.45	0.51	0.82
	SW	0.29	0.46	0.58	0.88

In another experiment on first dataset is attempted to modify a value of reward parameter and penalty parameter. A result of this experiment viewed in Fig. (3).

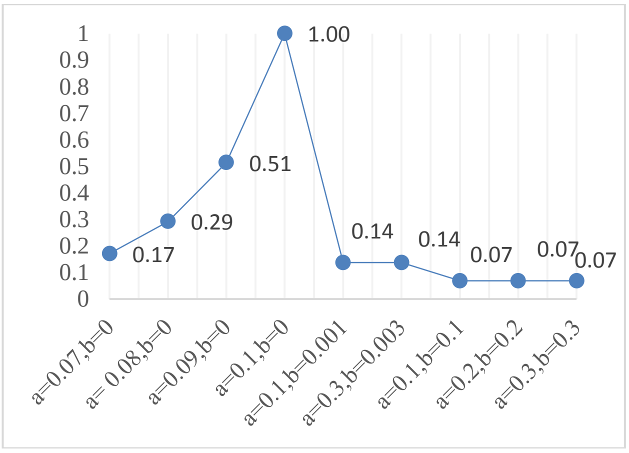


Fig. 9: Chart of change convergence rate with change reward and penalty parameter

Fig. (6) To Fig. (9) Show compare the accuracy of node classification between Multiple Ant Colony (MAC) [13], Factor Graph Model (FGM) [14] and our model with DLA on all datasets, with all local similarity measures. 80% of nodes are labeling in these experiments.

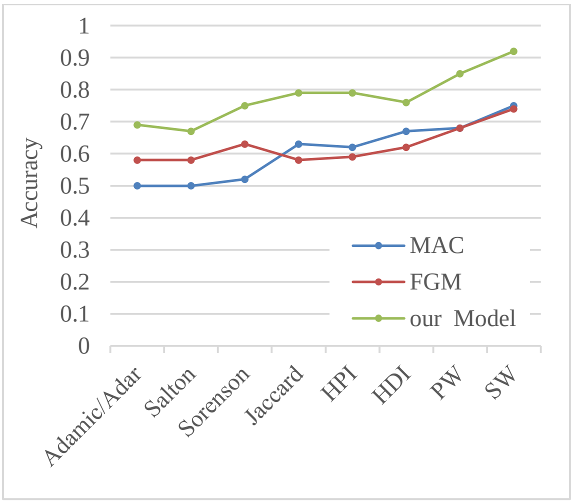


Fig. 10: Acuracy of node classification on Organizational Networks

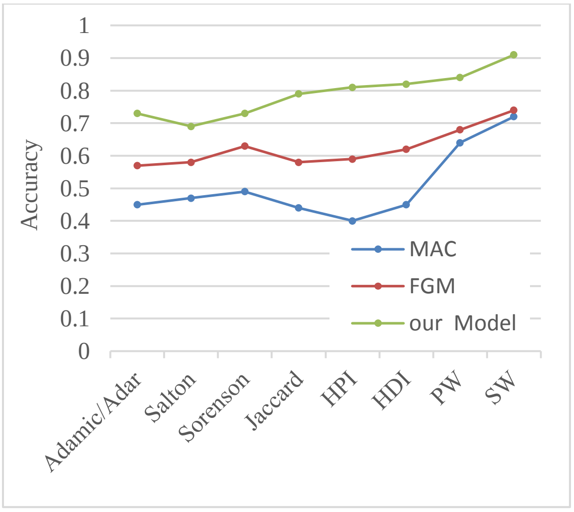


Fig. 11: Accuracy of node classification on Myspace

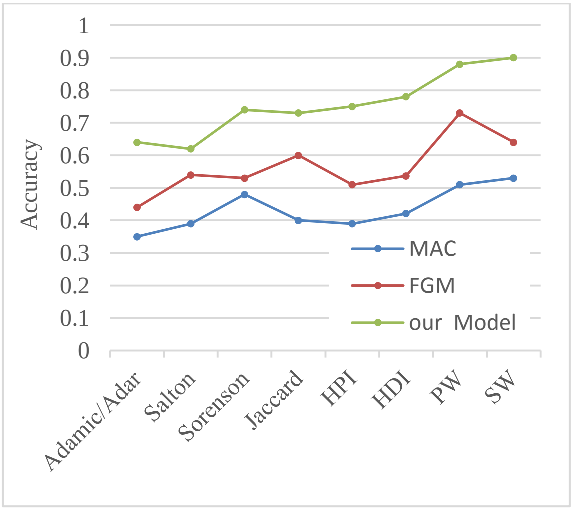


Fig. 12: Accuracy of node classification on Netlog

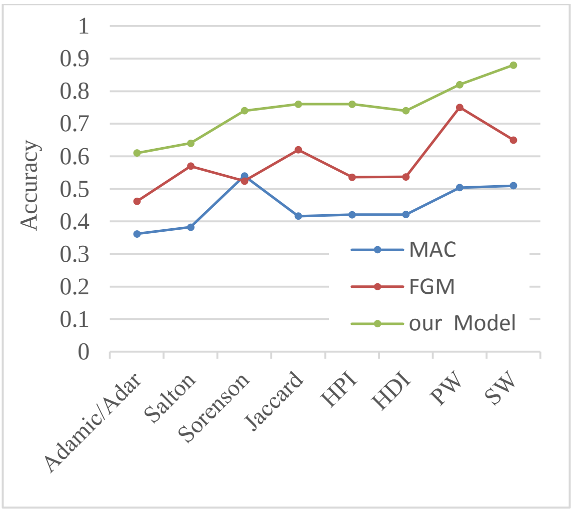


Fig. 13: Accuracy of node classification on Flickr

6. Conclusions

According to the result, the best of Local Similarity Measures is edge weight, and poor are Adamic/Adar. According to the Fig. (2) Convergence, rate is come down in L_{RP} and $L_{R\varepsilon P}$ .it shows that suboptimal behavior of learning automata in real environment preferred to optimal behavior. Also in L_{RI} the optimal value of a reward parameter is equal to 0.1. Convergence rate is come down with decrease or increase reward parameter. On an accuracy score, our model is more than better than others.

References

[1] J. Neville and D. Jensen, "Iterative classification in relational data," in Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data, 2000, pp. 13-20.
[2] S. A. Macskassy and F. Provost, "A simple relational classifier," DTIC Document2003.
[3] S. Bhagat, I. Rozenbaum, and G. Cormode, "Applying link-based classification to label blogs," in Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, 2007, pp. 92-101.
[4] S. Chakrabarti, B. E. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," ed: Google Patents, 2002.
[5] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in ICML, 2003, pp. 912-919.
[6] M. S. T. Jaakkola and M. Szummer, "Partially labeled classification with Markov random walks," Advances in neural information processing systems (NIPS), vol. 14, pp. 945-952, 2002.
[7] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," Proceedings of the VLDB Endowment, vol. 2, pp. 718-729, 2009.
[8] J. Kleinberg and E. Tardos, "Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields," Journal of theACM (JACM), vol. 49, pp. 616-639, 2002.
[9] F. McSherry, "Spectral partitioning of random graphs," in Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, 2001, pp. 529-537.
[10] A.B.Goldberg, X. Zhu, and S. J. Wright, "Dissimilarity in graph-based semi-supervised classification," in International Conference on Artificial Intelligence and Statistics, 2007, pp. 155-162.
[11] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in Proceedings of the 19th international conference on World wide web, 2010, pp. 641-650.
[12] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in Proceedings of the third ACM international conference on Web search and data mining, 2010, pp. 241-250.
[13] X. Xu, L. Lu, P. He, Y. Ma, Q. Chen, and L. Chen, "Semi-supervised classification with multiple ants maximal spanning tree," in Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on, 2013, pp. 315-320.
[14] H. Xu, Y. Yang, L. Wang, and W. Liu, "Node classification in social network via a factor graph model," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2013, pp. 213-224.
[15] H. Beigy and M. R. Meybodi, "Utilizing distributed learning automata to solve stochastic shortest path problems," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 14, pp. 591-615, 2006.
[16] M. Al Hasan and M. J. Zaki, "A survey of link prediction in social networks," in Social network data analytics, ed: Springer, 2011, pp. 243-275.
[17] C. A. Bliss, M. R. Frank, C. M. Danforth, and P. S. Dodds, "An evolutionary algorithm approach to link prediction in dynamic social networks," Journal of Computational Science, vol. 5, pp. 750-764, 2014.
[18] L. Adamic and E. Adar, "How to search a social network," Social networks, vol. 27, pp. 187-203, 2005.
[19] Y.-X. Zhu, L. Lü, Q.-M. Zhang, and T. Zhou, "Uncovering missing links with cold ends," Physica A: Statistical Mechanics and its Applications, vol. 391, pp. 5769-5778, 2012.

[20] W. Cukierski, B. Hamner, and B. Yang, "Graph-based features for supervised link prediction," in Neural Networks (IJCNN), The 2011 International Joint Conference on, 2011, pp. 1237-1244.

[21] E. Zheleva and L. Getoor, "To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles," in Proceedings of the 18th international conference on World wide web, 2009, pp. 531-540.

Ahmad RahnamaZadeh received the M.Sc degree in Software Engineering from Faculty of Electrical and Computer Engineering in Qazvin Islamic Azad University. He already received his B.Sc. of Computer Engineering, Hardware Engineering, at Islamic Azad University of Qazvin, Iran. His research interests include Social Network Analysis, Machine Learning and Graph Theory

Mohammad Reza Meybodi received the B.Sc and M.Sc degrees in Economics from Shahid Beheshti University, Tehran, Iran, in 1973 and 1977, respectively. He also received the M.Sc and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.

Masoud Taheri Kadkhoda received the M.Sc degree in Artificial Intelligence from Faculty of Electrical and Computer Engineering in Qazvin Islamic Azad University and received his B.Sc. of Computer Engineering, Software Engineering, at Mazandaran University of Science and Technology, Iran.