Learning Representations in Directed Networks

O. U. Ivanov[1] and S. O. Bartunov[2]

[1] Lomonosov Moscow State University, Russia
tigvarts@gmail.com,
[2] National Research University Higher School of Economics (HSE), Moscow, Russia
Computational Center of the Russian Academy of Sciences, Moscow, Russia
sbos@sbos.in

Abstract. We propose a probabilistic model for learning continuous vector representations of nodes in directed networks. These representations could be used as high quality features describing nodes in a graph and implicitly encoding global network structure. The usefulness of the representations is demonstrated on link prediction and graph visualization tasks. Using representations learned by our method allows to obtain results comparable to state of the art methods on link prediction while requires much less computational resources. We develop an efficient online learning algorithm which makes it possible to learn representations from large and non-stationary graphs. It takes less than a day on a commodity computer to learn high quality vectors on LiveJournal friendship graph consisting of 4.8 million nodes and 68 million links and the reasonable quality of representations can be obtained much faster.

Keywords: representation learning, graph embedding, link prediction, graph visualization, social network analysis.

1 Introduction

Graph data are ubiquitous: online social networks, web pages connected by hyperlinks, peer-to-peer file sharing networks and even protein interactions are naturally represented as graphs. However, it may be difficult to make predictions about objects organized in a graph assuming that predictions about connected objects are dependent since this may require to analyze the whole network. While it is possible to solve such problems using the framework of probabilistic graphical models [1], it may be non-trivial to derive efficient learning and inference algorithms for a particular model and exact inference is often intractable.

At the same time, machine learning methods that do not consider *structured* (that is, dependent) input or output space such as support vector machines, decision trees or feedforward neural networks became widely used. The independence of predictions for a collection of objects is a common assumption allowing for fast and even parallel learning algorithms.

Since it may be hard to work with raw data, objects are usually represented by a set of real numbers called *features*. There are many ways in which raw data

could be transformed into their feature representation suitable for learning algorithm and the choice of representation is crucial for prediction performance. The classic approach to machine learning involves hand-designed features prepared by domain experts with prediction problem of interest in mind. This paper address the problem of automatic feature extraction for nodes in a directed graph which could be used for independent predictions on a graph in order to overcome the limitations of non-structured learning algorithms.

In last decade there is growing interest in machine learning community to *representation learning* methods which allow to automatically extract useful features from complex objects such as images [2] and sound data [3]. While state of the art representation learning methods are usually intended to learn distributed representations with deep neural architectures, recently two lightweight shallow architectures were proposed for learning distributed word representations named Skip-gram and Continuous bag of words [4]. Despite much more simple formulations these two models could be trained much faster than recurrent neural network and outperform it in several tasks involving learned representations.

Motivated by success of these models we propose shallow bilinear model for learning representations of nodes in a graph. We show that using representations of relatively small dimensionality (about 30) it is possible to not just to recover the original graph, but also to accurately predict future links. Since learned low-dimensional vectors implicitly encode positions of nodes in the global structure of the network they may serve as high-quality input features in various data mining tasks.

2 Bilinear Link Model

Consider a directed network $G = (V, E)$, where V is the set of nodes and $E = \{(u_i, v_i)\}_{i=1}^{|E|} \subseteq V \times V$ is the set of links. We denote $d_+(u)$ as the number of outgoing links for node u and $d_-(u)$ as the number of ingoing links for node u.

The most obvious way to numerically represent node u is to encode its local connectivity information as binary vector r_u such that $r_{uv} = 1$ if $(u, v) \in E$ and $r_{uv} = 0$ otherwise. We may also consider a twice larger representation that accounts for incoming links as well. While such naive representation allows to reconstruct the network without loss of information, it has two important limitations which make it not very useful for a number of applications. First, it accounts only for *local* information and ignores *global* structure of the network which is also very important. For example, two nodes may not share any connections at all but still be considered close as belonging to the same structural component of the network. The second limitation is that length of the representation depends linearly on the number of nodes V which may be not practical for large networks with hundreds millions of nodes.

Below we describe our method for learning rich representations which overcomes these limitations by simultaneously compressing the graph and learning global interactions between nodes. We define the probabilistic Bilinear Link

Model (BLM) which explains local connections in the network by latent representations of the nodes. Thus, we associate each node u in the graph with its *input* and *output* representations which we denote as $In_u \in \mathbb{R}^D$ and $Out_u \in \mathbb{R}^D$, where D is the dimensionality of the latent space. Further we denote the set of all representations as $\theta = \{(In_u, Out_u)\}_{u \in V}$.

We assign the probability to each link (u, v) with source node u fixed according to the following bilinear softmax model:

$$p(v|u, \theta) = \frac{\exp(In_u^T Out_v)}{\sum_{w \in V} \exp(In_u^T Out_w)} \tag{1}$$

We maintain two representations for each node to explicitly express the orientation of a link by using corresponding representations.

In many applications it is required to calculate joint link probability. We may express it as $p(u, v|\theta) = p(u)p(v|u, \theta)$ and use maximum likelihood principle to estimate θ and $p(u)$:

$$J(\theta) = \sum_{i=1}^{|E|} \log p(u_i, v_i|\theta) = \sum_{i=1}^{|E|} \log p(v_i|u_i, \theta) + \sum_{i=1}^{|E|} \log p(u_i) \to \max_{\theta, p(u)} \tag{2}$$

We can show that maximum is attained when $p(u) = \frac{d_+(u)}{\sum\limits_{w \in V} d_+(w)} = \frac{d_+(u)}{|E|}$.

Unfortunately both evaluating $p(v|u, \theta)$ and computing the corresponding likelihood gradient requires normalizing over the entire network and thus (2) cannot be optimized efficiently.

The full gradient of (2) can be computed in $O(|V|^2 D)$ time and the $|V|^2$ term makes gradient descent method unacceptable for working with big data. Although we could employ stochastic gradient optimization and follow the direction of stochastic gradient estimated by a single randomly chosen link, each iteration will result into $O(|V|D)$ cost and the epoch (that is, pass over all links) requires $O(|E||V|D)$ time which is unacceptable for training on large graphs.

The most computationally expensive part is connected with normalization (denominator in (1)). Recently, noise contrastive estimation (NCE), a method for training unnormalized probabilistic models has been proposed [5]. The feature of this method is estimation a normalizing parameter $Z_u = \ln \sum_{w \in V} \exp(In_u^T Out_w)$ not any more as a function of θ but as an additional parameter of the model.

Further we show how to optimize the bilinear link model with NCE. We use an unnormalized model

$$p_{NCE}(v|u, \alpha) = \exp(In_u^T Out_v - Z_u)$$

as an estimation of $p(v|u, \theta)$, where $\alpha = \{(In_u, Out_u)\}_{u \in V}$, $Z_u \in \mathbb{R}$. Thus,

$$p_{NCE}(u, v|\alpha) = p(u)p_{NCE}(v|u, \alpha) = p(u) \exp(In_u^T Out_v - Z_u)$$

NCE optimizes the following logistic regression objective which separates objects from true data distribution $p_d(u, v)$ and samples from some *noise distribution* $p_n(u, v)$:

$$L_m(u, v, \alpha) = \ln \frac{p_{NCE}(v_i|u_i, \alpha)}{p_{NCE}(v_i|u_i, \alpha) + \nu p_n(\tilde{v}_i)}, L_n(u, v, \alpha) = \ln \frac{\nu p_n(\tilde{v}_i)}{p_{NCE}(\tilde{v}_i|\tilde{u}_i, \alpha) + \nu p_n(\tilde{v}_i)},$$

$$J_{NCE}(\alpha) = \frac{1}{|E|}\left(\sum_{i=1}^{|E|} L_m(u_i, v_i, \alpha) + \sum_{i=1}^{\nu|E|} L_n(\tilde{u}_i, \tilde{v}_i, \alpha)\right) \to \max_{\alpha} \qquad (3)$$

The choice of $p_n(u, v)$ is almost arbitrary, the most important requirement is that p_n should be nonzero whenever p_d is nonzero. We denote a set of edges generated from p_n as $\{(\tilde{u}_i, \tilde{v}_i)\}$, where $i \in \{1, \ldots, \nu|E|\}$, $\nu \in \mathbb{N}$, so there are ν times more noise samples than true data. In our study we used $p_n(u, v) = p(u)p_n(v)$, where $p_n(v)$ is a noise distribution for fixed source node, $p_n(v) \neq 0$ for all $v \in V$.

It was shown in [5] that optimization problem $J_{NCE}(\alpha) \to \max_{\alpha}$ leads to correct estimation of parameters α. In the limit (when amount of true data goes to infinity) the global maximum of J_{NCE} attains at such α^* that $p(u, v|\alpha^*) = p_d(u, v)$, where p_d is the true distribution (assuming that such α^* actually exists).

We can optimize $J_{NCE}(\alpha)$ (3) using stochastic gradient ascent. This allows us to handle online setting and the method is applicable for non-stationary graphs. On each iteration of the optimization we follow the direction of stochastic gradient estimated by single link. The computation complexity of each iteration is $O(|D|)$, so one epoch requires $O(|E|\nu D)$ time. This is much faster than $O(|E||V|D)$ for original model.

The model may tend to divergence. We can show that for $D = |V|$ optimal value of θ tends to infinity. We approach this problem by introducing regularizer $R(\alpha) = \sum_{u \in V} \left((\nu + 1)d_+(u)||In_u||_2^2 + (\nu + 1)d_-(u)||Out_u||_2^2\right)$. In this work we use weighted L_2-regularizer, because it has the property of isotropy for In_u and Out_u.

$$J_{NCE,R}(\alpha) = J_{NCE}(\alpha) + \gamma R(\alpha) \qquad (4)$$

This kind of regularizer puts hub nodes closer to the center of latent space. So on one hand the distribution on a target for link with a source fixed in a hub is less peaked. On the other hand a hub becomes a more likely target for all nodes in the network.

Denoting $\{\tilde{v}_{i,j}\}$ as a set of nodes generated from $p_n(v)$. Since $p(\tilde{u}) = p(u)$ we optimize the following function instead of $J_{NCE,R}(\alpha)$ (4) to simplify the implementation.

$$\hat{J}_{NCE,R}(\alpha) \quad = \quad \frac{1}{|E|}\left(\sum_{i=1}^{|E|}\left[L_m(u_i, v_i, \alpha) + \sum_{j=1}^{\nu} L_n(u_i, \tilde{v}_{i,j}, \alpha)\right]\right) \quad + \quad \gamma R(\alpha) \qquad (5)$$

One can see that if $d_+(u) = 0$, then In_u would never be changed during optimization, while Out_v is learned for every v because $p_n(v) \neq 0$. Therefore, Out representations are informative for all nodes, unlike In representations. Thus output vectors are more suitable as node features.

3 Related work

The classic approach to the problem of transforming nodes in a graph into real vectors involves projection on eigenvectors of graph Laplacian [6]. This is shown

to minimize the squared euclidean distance between representations of adjacent nodes. Such technique is also known as graph PCA since it preserves maximum variance in terms of euclidean commute time distance [7]. Despite these interesting theoretical properties spectral projections as feature vectors have certain disadvantages as we show in our experiments on visualization. In particular, while dissimilar nodes have large distances between corresponding spectral projections similar ones are located very close to each other making it hard to distinguish between them.

The important distinction between graph PCA and BLM is that our method is intended to work on directed networks while most results about spectral properties of graph Laplacian were obtained for undirected graphs. There are several developments on analysis of directed graph Laplacians [8,9] but no straightforward way to obtain vector representations using it is available for our best knowledge.

Recently an alternative approach for learning node features based on factorization of adjacency matrix [10] was proposed. It is similar to our method in its bilinear formulation, but in contrast to BLM lacks for probabilistic formulation, i.e. no direct way to obtain link probability is provided. On the other hand, this approach could be straightforwardly adapted to the case of undirected graphs while BLM is not designed for this case.

4 Experiments

4.1 Implementation details

To train models on huge networks, we use asynchronous parallel stochastic gradient ascent. The model was stored in the shared memory and optimized by several workers concurrently with no synchronization between them. Such approach was recently justified [11] by the fact that stochastic gradient estimates are usually sparse and concurrent updates rarely conflict. Moreover, any synchronization would affect the performance with almost no gain in test performance.

In our experiments we used constant learning rate 0.01, but using advanced learning rate schedules such as AdaGrad [12] should significantly increase speed of convergence and slightly increase the representation quality.

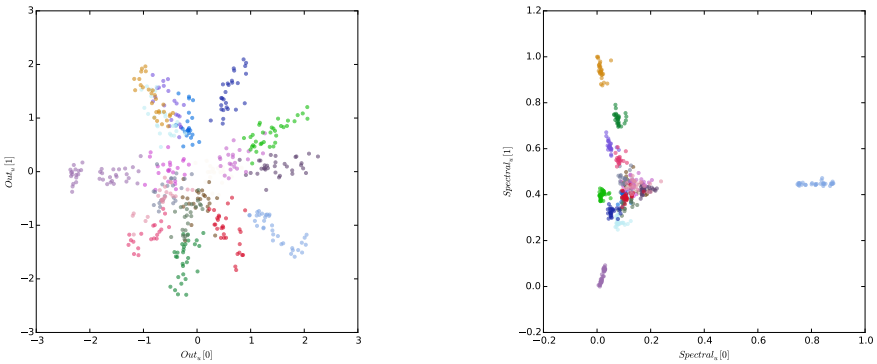
Representation vectors were initialized with small random numbers in a way that $\mathbb{E}In_u^T Out_v = 0$ and $\mathbb{E}Z_u = \ln|V|$. Regularization constant in (4) was set to $\gamma = 0.003$. We used noise distribution $p_n(v) = \frac{d_+(v)+d_-(v)}{2|E|}$ and ratio of noise links $\nu = 25$. We performed 200 epochs of stochastic gradient ascent on each dataset and as one can see in the experiments it was enough to converge.

For experiments we used C++ realization of the above algorithm with data representations stored in std::vectors and pthreads library for parallel computing.

4.2 Graph visualization

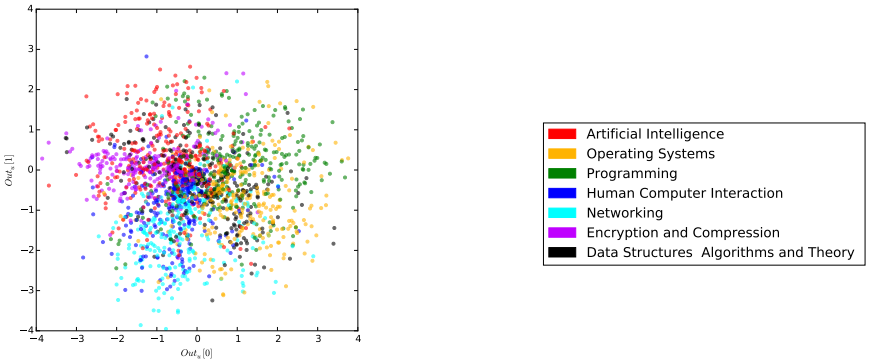
In this section we demonstrate the efficiency of our method for graph visualization task. We use our algorithm for learning 2-dimensional vectors which could

be used as node projections. As it was stated above, *Out* vectors are better at representing nodes so further we use them for visualization. We first conduct



BLM output vectors Spectral layout

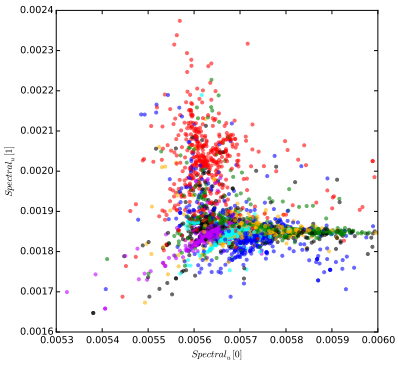
Fig. 1. Visualization of randomly generated scale-free network



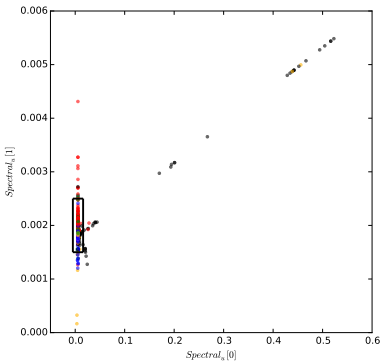
BLM output vectors

Fig. 2. Visualization of Cora citation network communities [13]

an experiment on randomly generated scale-free network. We used well known random graph generator [15] which is a standard tool for evaluation of various

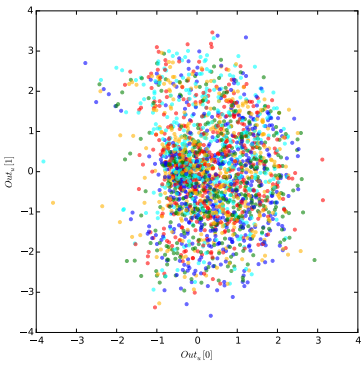


Spectral layout (zoom of the selected area on the right figure)

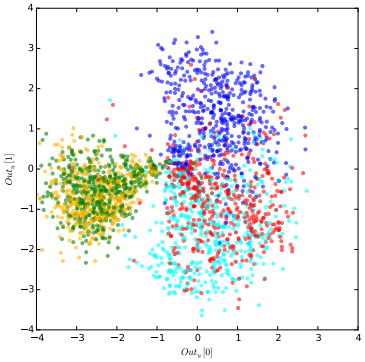


Spectral layout (all nodes)

Fig. 3. Visualization of Cora citation network communities [13]



The 5 largest communities



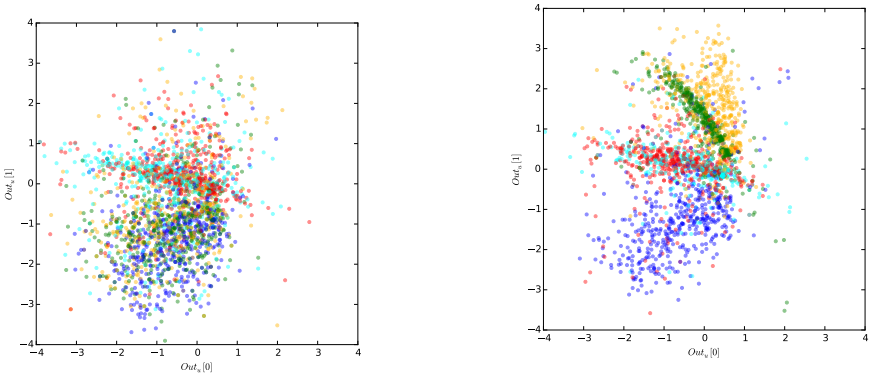
Communities 2001-st to 2005-th in descending order of size

Fig. 4. BLM visualization (not more than 500 random nodes from each community) of LiveJournal [14] communities

graph algorithms on synthetic data. We generated a 500 nodes graph with 19 non-intersecting a-priori known communities[3].

We visualized this graph on Figure 1 using BLM with $D = 2$ and projection on eigenvectors of graph Laplacian (we further refer to this method as *spectral layout*). Spectral layout proceeds only undirected graphs, so directed graphs are considered as undirected graphs for visualization. One can see that our model

[3] In this benchmark we used flags "-N 500 -k 15 -maxk 45 -mu 0.2 -t1 2 -t2 1 -minc 5 -maxc 30 -on 0 -om 0"



The 5 largest communities

Communities from 11-th to 15-th in descending order of size

Fig. 5. BLM visualization (not more than 500 random nodes from each community) of YouTube [14] communities

is suitable for visualising graphs: on one hand the communities compactness property is satisfied, on the other hand nodes from one community are distant enough to see the difference between them.

On a Figure 2 we can see our layout and on a Figure 3 we can see spectral layout for Cora citation network. Each paper in Cora is associated with a topic. Topics themselves are organized in a tree hierarchy. For this experiment we selected the 20 most popular topics. On the Figures 2 and 3 we joined intersected topics, such as Artificial Intelligence/Machine Learning/Neural Networks and Artificial Intelligence/Vision and Pattern Recognition in one label Artificial Intelligence and one color. For colors with number of nodes exceeding 500, we chose 500 random nodes of this color to visualize. We can see that in BLM model each topic corresponds to a certain direction (or few directions). The disadvantage of spectral layout is that its vectors do not reasonable describe the distance between two nodes. On the Figure 3 (note the scale of both axes!) we can see, that distances between pairs of black nodes are unevenly distributed and may be hundreds times more than distance between other topics and their diameters. Thus, spectral layout is hard to be used as reasonable feature vectors for graph data. However, spectral layout may be better for visual community detection if the communities are known.

On Figures 4 and 5 we visualized the largest communities for LiveJournal and YouTube social networks. We can see, that the visualization of the largest communities is not very informative. The reason is the largest communities are meant to be popular, so they may contain people of very different interests. So on Figures 4 and 5 one can mention a good separability of not so popular communities.

Above we used Lanczos method for computing spectral layouts but failed to compute eigenvectors of LiveJournal and YouTube graphs in a reasonable time because of large sizes of these graphs. In contrast to Lanczos method our stochastic algorithm has linear complexity in number of edges and could be stopped at any moment while sub-optimal representations would be still useful.

4.3 Link prediction

Since function (5) is associated with estimation of link probability, it is natural to use obtained representations in link prediction task.

In order to perform evaluation on this task we assume that each link prediction method returns a score for each unordered pair of nodes: the higher value means the more likely a link between them. Introducing a separator turns this method into a classifier, which shows if the probability of a link is higher than random. Area under the curve (AUC) is a standard metric to measure the quality of a classifier.

To assess a link prediction method quality the links of the network are randomly divided into two parts: observed links and missed links. Links not from network are called non-existing links. The observed links are a known for algorithm information, while no information about missed links is allowed to be used in prediction. So the task for a link prediction algorithm is to separate missed links and non-existing links.

Unfortunately, there are nearly $|V|^2$ non-existing links, so it is impossible to compute AUC analytically. To solve this problem, we can choose randomly one missed link, one non-existing link, and compare their scores. The AUC value can be estimated unbiased as follows:

$$AUC = \frac{n_{greater} + 0.5 n_{equal}}{n_{total}}$$

n_{total} is a number of such comparisons, $n_{greater}$ is number of comparisons where a missed link has a greater score, n_{equal} is a number of comparisons where both links has an equal score.

On the Figure 6 we can see the AUC measure estimation for citation cit-HepPh network [16] during the optimization process. Train set plot is AUC for separating observed links from the others. We can see it tends to 1, which means that obtained representations efficiently compress the network structure. Test set plot is AUC for separating missed links from non-existing links. "Fair" Z_u means that on each step for testing we use not a NCE estimation Z_u, but analytically computed $Z_u = \ln \sum_{w \in V} In_u^T Out_w$. We can see on the plot, that NCE estimates Z_u accurately.

Since we used constant learning rate, the convergence of the optimization can be assessed by this plot. Also it is clear that 200 epochs are enough to get high quality representations, but reasonable quality representations can be obtained much faster.

The dependence of train set AUC and test set AUC on dimension of representations for LiveJournal social network [16] after BLM learning is shown on

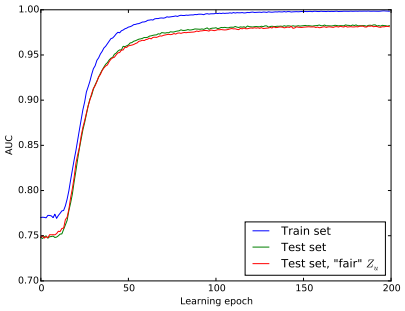


Fig. 6. AUC estimation on citation cit-HepPh network [16] during optimization

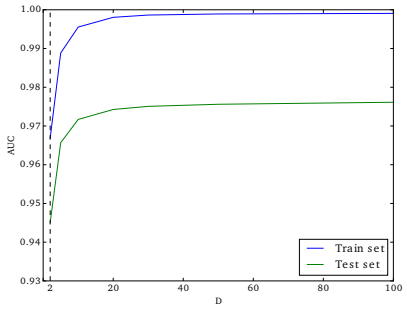


Fig. 7. Dependence of AUC on dimensionality of representations on LiveJournal [16]

the Figure 7. We see that the higher the dimension D, the smaller the increase in AUC measure. We assume $D = 30$ is large enough to capture most interactions in the network.

Baselines. We selected several representative baseline algorithms to compare with according to the survey of link prediction methods [17]. There are two methods for link prediction, namely local and global.

Local similarity indexes are methods that don't use any information about global network structure. They have nearly equal accuracy. We decided to use Jaccard metric for our test, because it is almost the easiest and rapidly computable algorithm to implement. Jaccard similarity metric between two nodes and is a ration of their common neighbours to total number of their neighbours.

Global similarity indexes use information about global network structure. According to the survey [17] random walks are the most accurate technique known for link prediction. The point in these algorithms is computing probability for a random walker, who starts from a fixed node and moves randomly, to appear in another fixed node. We included in the evaluation state of the art random walk methods namely local random walk (LRW) and superposed random walk (SRW) [18].

Results. In this paper we compare AUC measure of Jaccard similarity, LRW with 3 steps, SRW with 3 steps (it is the most popular number of steps according to the survey [17]), and our method with $D = 30$. We selected 5% of network links randomly as test set and the rest as train set available for algorithms. Results are presented in Table 2. One may see that representations obtained by our method solve link prediction task with a comparable to the state of the art quality. Experiments show that for soc-LiveJournal using $\nu = 10$ in BLM leads to the same results two times faster.

The results on soc-Pocek differs from the results on the other networks. The most methods degrade on this network, especially Jaccard. That can be explained

by the sparsity of the network: soc-Poces has average clustering coefficient 0.109, while average clustering coefficients of the other networks are from 0.274 to 0.597. The investigation of the dependence between the quality of link prediction algorithms and the sparsity of the network is a direction for further research, but it is out of scope of this paper.

Table 1. Link prediction methods performance. Time was measured for LiveJournal social network [16] on one core of Intel(R) Xeon(R) E5-2670 2.60GHz CPU. Time for BLM does not take into account the training effort.

	BLM	Jaccard	LRW (T steps)	SRW (T steps)
Score function evaluation cost	$O(D)$	$O(\frac{\lvert E\rvert}{\lvert V\rvert})$	$O(\lvert E\rvert T)$	$O(\lvert E\rvert T)$
Parameters for one evaluation	$D = 30$		$T = 3$	$T = 3$
Time for one evaluation, sec.	10^{-6}	$4.65 \cdot 10^{-5}$	3.13	3.13

As one can see in Table 1, LRW and SRW method are not fast enough to make more than 10^5 compares in AUC. So we estimated AUC with $n_{total} = 10^5$ and in the following table we truncated values to the last reliable digit.

Table 2. Link prediction, AUC

Dataset [16]	BLM(30)	Jaccard	LRW(3)	SRW(3)
soc-LiveJournal	0.975	0.938	0.986	0.985
soc-Pocek	0.978	0.850	0.966	0.967
web-Google	0.961	0.945	0.977	0.978
web-BerkStan	0.979	0.960	0.996	0.996
cit-HepPh	0.983	0.962	0.988	0.989

5 Discussion and future work

In this paper we described a novel scalable method for learning node representations in directed graphs. Initial results showed that our method is useful in graph analysis and allows to learn high-quality dense feature vectors which implicitly compress structure of the graph. Our study is still ongoing and we plan to investigate how representations learned by BLM could be used for various predictions on graphs such as community membership or estimating node attributes e.g. age or interests listed in a social network profile. Another direction of research is to consider other possible formulations of similarity between representations used in softmax equation (1). Finally, it would be interesting to adapt the model to undirected networks.

Acknowledgements. This work was supported by RFBR grant 14-01-31361.

References

1. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press (2009)
2. Krizhevsky, A., Hinton, G.E.: Using very deep autoencoders for content-based image retrieval. In: ESANN. (2011)
3. Graves, A., rahman Mohamed, A., Hinton, G.E.: Speech recognition with deep recurrent neural networks. CoRR **abs/1303.5778** (2013)
4. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. (2013) 3111–3119
5. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. Journal of Machine Learning Research **13**(1) (February 2012) 307–361
6. Chung, F.R.K.: Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92). American Mathematical Society (December 1996)
7. Saerens, M., Fouss, F., Yen, L., Dupont, P.: The principal components analysis of a graph, and its relationships to spectral clustering. In: Proceedings of the 15th European Conference on Machine Learning (ECML 2004). Lecture Notes in Artificial Intelligence, Springer-Verlag (2004) 371–383
8. Perrault-Joncas, D.C., Meila, M.: Directed graph embedding: an algorithm based on continuous limits of laplacian-type operators. In Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F.C.N., Weinberger, K.Q., eds.: NIPS. (2011) 990–998
9. Chung, F.: Laplacians and the cheeger inequality for directed graphs. Annals of Combinatorics **9** (2005) 1–19
10. Menon, A.K., Elkan, C.: Link prediction via matrix factorization. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II. (2011) 437–452
11. Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., eds.: Advances in Neural Information Processing Systems 24. Curran Associates, Inc. (2011) 693–701
12. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12** (July 2011) 2121–2159
13. : Cora citation network dataset – KONECT (October 2014)
14. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and Analysis of Online Social Networks. In: Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07), San Diego, CA (October 2007)
15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78** (Oct 2008) 046110
16. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (June 2014)
17. Lü, L., Zhou, T.: Link prediction in complex networks: A survey. Physica A Statistical Mechanics and its Applications **390** (March 2011) 1150–1170
18. Liu, W., Lü, L.: Link prediction based on local random walk. EPL (Europhysics Letters) **89**(5) (2010) 58007