Learning Word Vectors in Deep Walk Using Convolution

Iti Chaturvedi, Sandro Cavallari, Erik Cambria
School of Computer Science and Engineering
Nanyang Technological University
{iti,sandro001,cambria}@ntu.edu.sg

Vincent Zheng
Advanced Digital Sciences Center
vincent.zheng@adsc.com.sg

Abstract

Textual queries in networks such as Twitter can have more than one label, resulting in a multi-label classification problem. To reduce computational costs, a low-dimensional representation of a large network is learned that preserves proximity among nodes in the same community. Similar to sequences of words in a sentence, DeepWalk considers sequences of nodes in a shallow graph and clustering is done using hierarchical softmax in an unsupervised manner. In this paper, we generate network abstractions at different levels using deep convolutional neural networks. Since class labels of connected nodes in a network keep changing, we consider a fuzzy recurrent feedback controller to ensure robustness to noise.

Introduction

In the last few years, graphs have become popular data structures to model social networks like Facebook, Twitter and community based question answering systems (Fang et al. 2016). Textual data such as queries, web-pages, products and even users can have more than one label (Chen et al. 2017). For instance, a query such as Jaguar can have a label set containing Animals, Software and Automotive classes. Due to the huge dimension and the sparsity of these graphs, traditional machine learning algorithms struggle to perform well on this kind of data (Cambria, Wang, and White 2014).

Hence, some researchers proposed the use of community embedding techniques, which project the original network structure onto a low-dimensional latent/hidden vector space (Zheng et al. 2017). The learned vector space can be easily utilized by algorithms for tasks such as node classification or community detection. Similar, to sequence of words in a sentence, DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and LINE (Tang et al. 2015) consider a sequence of nodes in a graph connected by edges (Mikolov et al. 2013). Shallow learning in the form of hierarchical softmax or negative sampling is used to predict the dependencies among far away nodes in an unsupervised way. The resulting model appears capable of preserving correctly the second-order proximity, thus ensuring that nodes in the same community are closer in the embedded space.

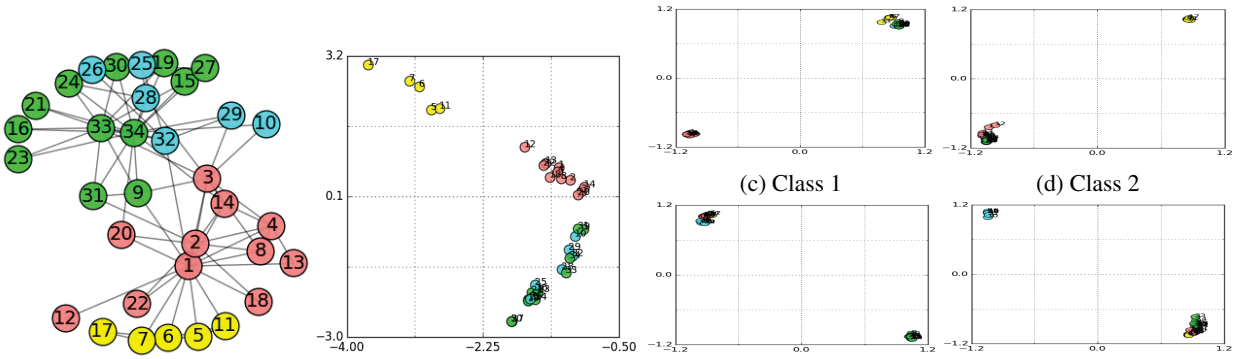
Fig. 1 illustrates a toy example where the well-known karate network (Zachary 1977) is projected onto a 2D space using the embeddings obtained from DeepWalk respect to our model. This is a non-linear problem, that can be effectively solved using several layers of neurons. In order to reduce computational costs, a multi-class problem can be decomposed into a number of binary problems knows as one-vs-rest. This approach can be extended to multi-label prediction by using the F1 evaluation metric to optimize the SVM decision threshold for imbalanced binary classifiers. Hence, given a node v_i, instead of learning a single embedding ϕ_i, in a traditional one-for-all way, we learn K different embeddings ϕ_{i_k} for each class k in our dataset.

Several authors have considered first- or second-order proximity in graphs, however only few works exploit a deep network to obtain a higher abstraction. Our work is inspired by the recent use of deep auto-encoders to learn graph embeddings in multi-class problems (Cao, Lu, and Xu 2016). In this paper, we argue that supervised learning can highly affect the final nodes embedding since can better separate nodes of different classes and highlight some communities structure. While nodes and node's neighborhoods are evident property of a graph, communities can have blurred boundaries and detecting them is challenging, for this reason we exploit the abstraction property of deep convolutional neural networks (DCNN).

Random walks are low complexity similarity measures ideal for large-scale network mining. A random walk is a stochastic process that starts with a root node and choses a random vertex from its neighborhood. In this way, the class labels of the nodes in a random walk will keep changing depending on the community. In this paper, we also introduce a fuzzy feedback controller, in order to model in a stable manner the changes in class labels for a sequence of connected nodes in social networks. The resulting model is referred to as fuzzy convolutional deep walk (FCDW).

Related Work and Contributions

Many applications such as on-line advertisements need to detect different type of users, hence a valuable network representation is mandatory (Majumder et al. 2017). A major challenge is that many real-world networks are sparse, with very few observed links among nodes resulting in poor classification accuracy.



(a) Karate network (b) DeepWalk embedding (c) Class 1 (d) Class 2 (e) Class 3 (f) Class 4

(g) DCNN embedding

Figure 1: Embedding of the karate network in a 2D vector space by different methods.

In (Cao, Lu, and Xu 2015), the authors present a model named GraRep, which applies matrix factorization to the graph's adjacency matrix for learning a new vertex representation. Node2Vec (Grover and Leskovec 2016) is a framework based on DeepWalk, but exploits a robust path sampling algorithm to obtain a more accurate embedding. It can be seen that these embedding models are shallow. However, since the underlying network structure is complex, shallow models cannot capture the highly non-linear network structure, resulting in sub-optimal network representations.

Recently, several authors have proposed deep models to learn graph embeddings. In (Wang, Cui, and Zhu 2016), the authors propose a semi-supervised deep neural network (DNN) that considers first-order and second-order proximity among nodes. The model is trained on each node and its neighborhood as obtained from the graph. In this way, the number of possible sequences grows exponentially, whereas we consider the low-dimensional latent representation of the network as a source for training our DCNN model. In this way, we are able to reduce the complexity of the model and avoid overfitting.

In (Niepert, Ahmed, and Kutzkov 2016), the authors applied 2D CNN for images to construct locally connected neighborhoods from the input graphs. Similar to the sequence of words in a sentence, the sequence of nodes in a graph are also unique from left to right and also from top to bottom. This method uses a sequence of nodes as input, transforming them into neighborhood graphs. Different graph structures are hence learned via convolution. While they consider 2D convolution to learn hubs and patterns in the graph, in our method we have focused on 1D convolution over latent low-dimensional vectors that can be used to detect communities. Lastly, in (Chang et al. 2015), the authors considered the scenario where nodes are of different types such as image and text. They proposed separate deep models for each type of data that is computationally expensive. Our method, on the other hand, makes use of latent vectors that can easily combine multimodal data (Poria et al. 2016).

The significance and contributions of the research work presented in this paper can be summarized as follows:

- We extend the 1D deep convolutional kernels in language model to learn the sequence of nodes in each community in a supervised manner.
- The DNN is able to generate network abstractions at different levels in complex social networks.
- Robustness of the model to noise is achieved using a fuzzy recurrent feedback controller.

To verify the effectiveness of FCDW in multi-labelling of nodes in large networks, we consider the BlogCatalog and Flickr datasets. The first is a social network of bloggers and is an image hosting website. The remainder of the paper is organized as follows: next section provides some preliminaries; following, a section introduces the FCDW model for multi-label classification of large social networks; next, an experiment section validates FCDW on real-world benchmark datasets; finally, a section concludes the paper.

Preliminaries

We briefly review the theoretical concepts necessary to comprehend the presently discussed work. We begin with a description of multi-label classification in graphs. We next transform the problem into a language model and show that it can be solved effectively by using DCNNs.

Multi-label Classification

Graph embedding aims to learn an embedding of d dimensions $\phi_i \in \mathrm{R}^d$ for each node $v_i \in V$, where V is the node set of a graph $G = (V, E)$ and E is the edge set. We can define node embedding as a set of random variables for a distribution characterizing how neighborhood's nodes are distributed in the low-dimensional space. To learn the node embedding ϕ_i's of a node v_i, we consider the second-order proximity between v_i and the context of its nodes' neighborhoods $\phi_j^{'}$ as well as the class labels of v_i. Let n denote the number of nodes in the graph, $m = n$ is the number of features given by the adjacency matrix defined by edge set E, and K is the number of class labels. Hence, $\mathbf{X} \in \mathrm{R}^{n \times n}$ is the training embedding matrix, and $\mathbf{Y} \in \{0, 1\}^{n \times K}$ is the corresponding label matrix. Given a positive integer d,

the multi-label problem seeks to find an embedding ϕ with exactly $d << n$ columns such that the error in classification is minimized. In this paper, we transform the multi-label problem to a multi-class problem as described in (Cherman, Monard, and Metz 2011). Following this procedure instead of learning a single embedding ϕ_i, we proposed to learn K different embedding ϕ_{i_k} for each class k. In this way, we are able to maximize the separation between nodes of different classes.

Language Model

We denote a random walk rooted at vertex v_i as a stochastic process with random variables $W^1_{v_i}, W^2_{v_i}, \ldots, W^k_{v_i}$ such that the $W^{k+1}_{v_i}$ is a vertex chosen at random from the neighbors of vertex v_k. We can rephrase the problem of estimating the class of a vertex given all previous vertices in a random walk as a language model. In a language model we consider a sequence of words $(w_0, w_1 \ldots, w_n)$ appearing in a corpus and try to maximize the probability $p(w_n|w_0, w_1, \ldots, w_n - 1)$ over the whole training corpus. However, our goal is to learn a latent representation, not only a probability distribution of node co-occurrences, and so we introduce a mapping function $\phi : v \in V, V \in \mathrm{R}^{|V|\times d}$. This mapping ϕ represents the latent social representations associated with each vertex v in the graph. Furthermore, the context is composed of words appearing to the left and the right side of a word resulting in a skip-gram. The optimization problem then becomes:

$$\min_\phi - \log p(v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}|\phi(v_i)) \quad (1)$$

In the next section we show that, this problem can be effectively solved using DCNNs.

Deep Neural Networks

A DNN can be viewed as a hierarchy of unsupervised models called restricted Boltzmann machines (RBMs), where each hidden layer serves as the visible layer for the next RBM. Each RBM is a bipartite graph comprising two layers of neurons: a visible and a hidden layer; where the connections among neurons in the same layer are not allowed.

To train such a multi-layer system, we must compute the gradient of the total energy function E_d with respect to the weights in all layers. To learn such weights and maximize the global energy function, the approximate maximum likelihood contrastive divergence approach can be used. This method employs each training sample to initialize the visible layer. Next, it uses the Gibbs sampling algorithm to update the hidden layer and then reconstruct the visible layer in succession, until convergence. As an example, here we use a logistic regression model to learn the binary hidden neurons, with each visible unit assumed to be a sample from a normal distribution. The continuous state $\hat{h}_j$ of the hidden neuron j, with bias b_j, is a weighted sum over all continuous visible nodes $\boldsymbol{v}^{'}$ and is given by:

$$\hat{h}_j = b_j + \sum_i v_i^{'} w_{ij}, \quad (2)$$

where w_{ij} is the connection weight to hidden neuron j from visible node $v_i^{'}$.

The binary state h_j of the hidden neuron can be defined by a sigmoid activation function:

$$h_j = \frac{1}{1 + e^{-\hat{h}_j}}, \quad (3)$$

Similarly, in the next iteration, the continuous state of each visible node $v_i^{'}$ is reconstructed. Lastly, the weights are updated as the difference between the original and reconstructed visible layer labelled as the vector $\boldsymbol{v}_{recon}$ using:

$$\triangle w_{ij} = \alpha(< v_i^{'} h_j >_{data} - < v_i^{'} h_j >_{recon}), \quad (4)$$

where α is the learning rate and $< v_i^{'} h_j >$ is the expected frequency with which visible unit i and hidden unit j are active together when the visible vectors are sampled from the training set and the hidden units are determined by (2). Finally, the energy of a DNN can be determined in the final layer using:

$$\mathrm{E}_d = -\sum_{i,j} v_i^{'} h_j w_{ij}, \quad (5)$$

To extend the DNN to a DCNN, we simply partition the hidden layer into Z groups. Each of the Z groups is associated with a $n_x \times n_y$ filter where n_x is the width of the kernel and n_y is the height of the kernel. Assume that, the input has dimension $\mathrm{L_x} \times \mathrm{L_y}$, then the convolution will result in a hidden layer of Z groups each of dimension $(\mathrm{L_x} - n_x + 1) \times (\mathrm{L_y} - n_y + 1)$. These learned kernel weights are shared among all hidden units in a particular group. The energy function of layer l is now a sum over the energy of individual blocks given by:

$$\mathrm{E}^l = -\sum_{z=1}^{\mathrm{Z}} \sum_{i,j}^{(\mathrm{L_x}-n_x+1),(\mathrm{L_y}-n_y+1)} \sum_{r,s}^{n_x,n_y} v_{i+r-1,j+s-1}^{'} h_{ij}^z w_{rs}^l. \quad (6)$$

Hence, each layer of a DCNN is referred to as a convolution RBM (CRBM). In this paper, since there is no special interdependence between 2 node embedding ϕ_{i_k} and ϕ_{j_k}, we set n_y to 1. In such a model, the lower layers learn abstract concepts and the higher layers learn complex features for clusters of nodes.

Fuzzy Convolutional Deep Walk

In this section, we describe the complete framework for using DCNN for multi-label classification of nodes in social networks. We introduce a layer of fuzzy neural network and a feedback controller to stabilize the changes of class labels in a sequence of connected nodes (Xing, Cambria, and Zou 2017).

Fuzzy Feedback Controller

Stability analysis is an important issue for fuzzy control systems. Recently, linear matrix inequalities (LMI) are used to reduce stability analysis in fuzzy systems with time delays (Wang, Tanaka, and Griffin 1996).

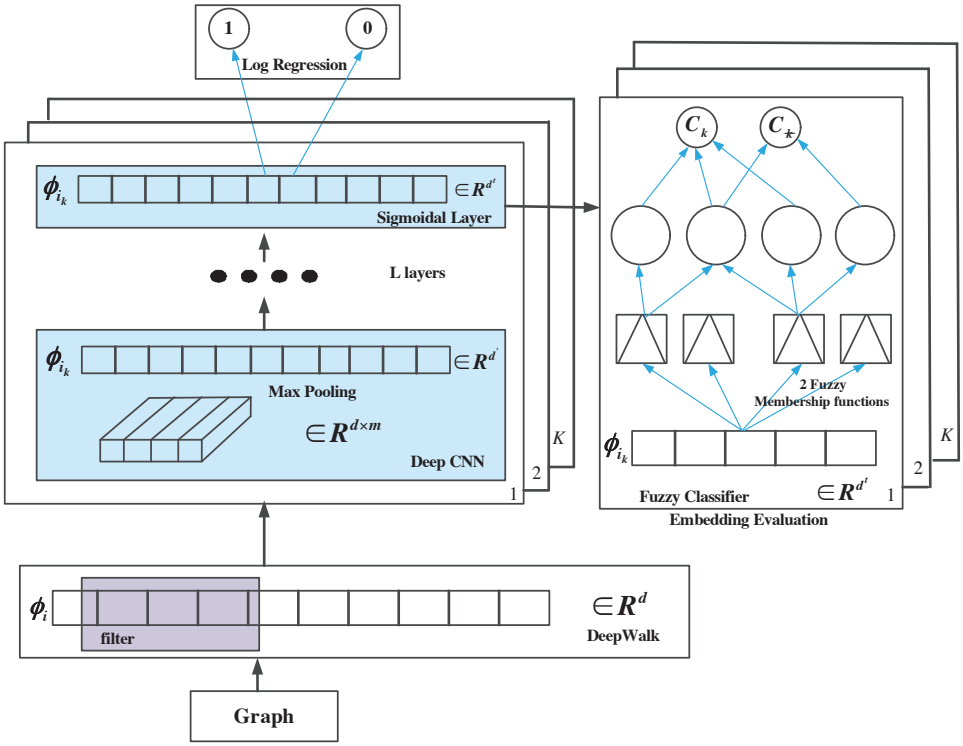


Figure 2: The state space of FCDW for learning communities in social networks.

A fuzzy system can be defined by IF-THEN rules, which locally represent linear input-output relations of a system:

$$\begin{aligned} &\text{IF } x_1(t) \text{ is } M_{i1} \ldots x_n(t) \text{ is } M_{in} \\ &\text{THEN } \boldsymbol{x}(t+1) = A_i \boldsymbol{x}(t) + B_i \boldsymbol{u}(t) \end{aligned} \tag{7}$$

where i is the rule index and t is the time point in a sequence. For example, Fig. 3 shows two membership functions for Bloggers in the interest groups 'Business' and 'Music'. As M_{i1} goes down the value of M_{i2} increases.

Eq (7) gives the fuzzy classifier output $x(t+1)$ as a function of the output $x(t)$ at previous time point t. Stability is achieved by designing an additional gain matrix $\boldsymbol{F}$ that modifies the input data such that it is robust to small perturbations. In this paper, the problem is simplified to a linear matrix inequality (LMI) using the Schur complement lemma as follows:

$$\begin{bmatrix} \boldsymbol{Q} & (\boldsymbol{A_i Q} - \boldsymbol{B_i K})^T \\ (\boldsymbol{A_i Q} - \boldsymbol{B_i K}) & \boldsymbol{Q} \end{bmatrix} > 0 \tag{8}$$

where $\boldsymbol{F} = \boldsymbol{K Q}^{-1}$ and there exists $\boldsymbol{P} > 0$. This LMI can be easily solved using convex optimization. Proof is given in (Wang, Tanaka, and Griffin 1996).

Fuzzy Deep Walk Framework

Fig. 2 Illustrates the state space of FCDW for learning communities in social networks. We learn low-dimensional latent embeddings ϕ_{i_k} from random walk sequences using skip-gram model. The new latent dimensions d are used to train a DCNN. The DCNN is trained to assign each node v_i to a given class which represents a community. Given this training procedure, the network learns $d^{'}$ kernels or filters in each layer that correspond to the different communities representations. Lastly, we consider d^t fuzzy recurrent neurons with feedback for stable learning of sequences of nodes. There are 2 fuzzy membership functions at each output neuron. The output layer corresponds to different community labels.

Experiments and Results

In this section, we evaluate the proposed FCDW on two real-world multi-label datasets known as BlogCatalog and Flickr (Tang and Liu 2009). We compare our methods against three other baseline methods: EdgeCluster (EC)(Tang and Liu 2009), SpectralClustering (SC)(Tang and Liu 2011) and DeepWalk. We begin with the details of F1-macro evaluation metric, next, we describe the dataset and evaluate the results. Lastly, we visualize the communities learned by FCDW.

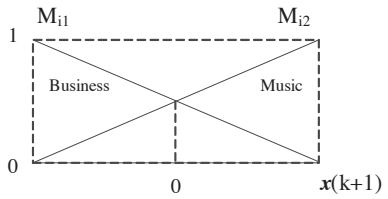


Figure 3: Membership functions for Bloggers of two different interest Groups

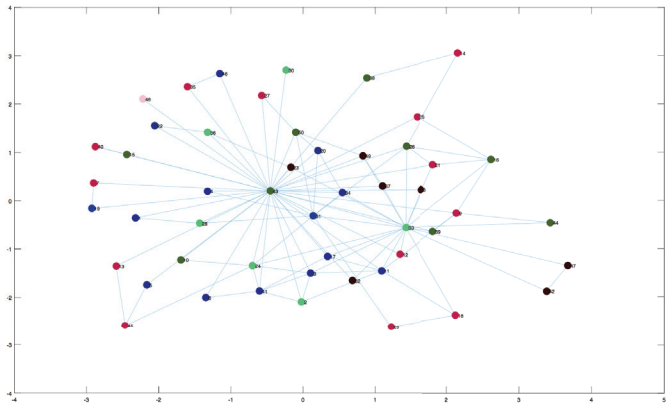


Figure 4: BlogCatalog community detection

Evaluation Measure and Datasets

We consider the Macro-F1 score, that is F1 averaged over categories as defined in (Perozzi, Al-Rfou, and Skiena 2014).

$$Macro - F1 = \frac{1}{K}\sum_{k=1}^{K} F_1^k \tag{9}$$

For a category C_k, the precision P^k and the recall R^k are calculated as,

$$P^k = \frac{\sum_{i=1}^{N} y_i^k \hat{y}_i^k}{\sum_{i=1}^{N} \hat{y}_i^k}, R^k = \frac{\sum_{i=1}^{N} y_i^k \hat{y}_i^k}{\sum_{i=1}^{N} y_i^k} \tag{10}$$

where $y_i, \hat{y}_i \in \{0,1\}^K$ are the true and predicted labels respectively. Then the F1 measure, defined as the harmonic mean of precision and recall, is computed as follows:

$$F_1^k = \frac{2P^k R^k}{P^k + R^k} \tag{11}$$

BlogCatalog BlogCatalog is a network of social relationships provided by blogger authors. The nodes can belong to one of the 39 categories. There are a total of 10,312 nodes connected by 333,983 edges. The labels represent the topic categories provided by authors. In this experiment we consider the BlogCatalog network with 80% training data.

Flickr Flickr is an image/video hosting website and online community. The crawled graph represents the friendship network and the group membership. It results in a network composed by 80,513 nodes and 5,899,882 edges, containing 195 communities. Given the dimension of the network, in this case, we use only 8% of the data as training set.

Tuning of Hyper-parameters We consider a validation set to determine the hyper-parameters. The number of hidden neurons in each layer and the number of layers is gradually increased until performance saturates due to overfitting. We consider 1D convolutional kernel of width (k=3/4/5) and up-to 7 layers. Similarly, the number of neurons and membership functions in the fuzzy classifier were set to 10 and 2 based on highest accuracy and speed. We used Theano based stochastic gradient descent for deep learning and Simulink to implement the fuzzy classifier.

Evaluation

In order to measure the improvement provided by using FCDW alone as compared to DeepWalk, we first extracted the latent vectors from the graph using both methods. Next, we classified both datasets using neuro-fuzzy classifier and feedback controller. We were unable to get a meaningful result for the feedback controller on DeepWalk features, as the gradients are difficult to compute. On the other hand, DCNN is able to heuristically compute the gradients using contrastive divergence.

Table 1 shows the Macro-F1 when 128 latent dimensions, as predicted using DeepWalk for BlogCatalog and Flickr dataset. We can see that features learned via DCNN outperform DeepWalk. We can see that the percentage improvement increase significantly with size of social network. The final results are presented in Table 2. Compared to our baseline, FCDW outperforms DeepWalk by over 5% when we consider the Macro-F1 measure on BlogCatalog. The improvement is over 15% on the bigger network of Flickr.

Table 1: Macro-F1 by DeepWalk+Fuzzy and FCDW for BlogCatalog and Flickr dataset.

	DeepWalk+Fuzzy	FCDW
BlogCatalog	0.33	**0.34**
Flickr	0.32	**0.39**

Table 2: Macro-F1 by different models for classifying nodes in BlogCatalog and Flickr Dataset.

	EC	SC	DeepWalk	FCDW
BlogCatalog	0.23	0.31	0.28	**0.34±0.05**
Flickr	0.20	0.24	0.25	**0.39±0.06**

Communities Evaluation

In order to evaluate the ability of our algorithm of correctly separate the 2 classes in each embedding ϕ_{i_k}, we compute the normalized mutual information (NMI). To obtain this evaluation, we used the obtained embedding from each methods and then we fit it in a GMM clustering algorithm. The GMM is set to have only 2 components, one that represent the evaluated class at step k and one that represent all the other nodes.

Proceeding with this One-vs-Rest strategies, we are able to compute the NMI for each class and report the average among all the classes. Table 3 show how FCDW is able to preserve the community's structure and outperforming DeepWalk and LINE by a factor of 10 on BlogCatalog.

Table 3: NMI by different methods for evaluating the ability of detecting the different communities.

	DeepWalk	LINE	FCDW
BlogCatalog	0.00352	0.00346	**0.0318**
Flickr	0.0066	0.0067	**0.0081**

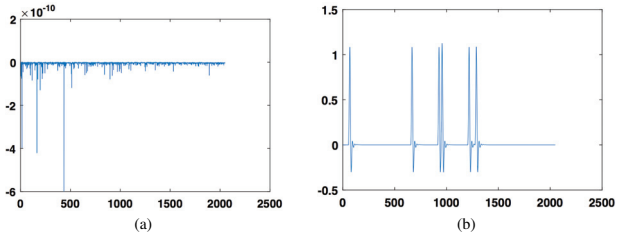


Figure 5: (a) The output class labels from neuro-fuzzy classifier (b) The output class label after processing with feedback controller

To give a better understanding of how FCDW is able to preserve the communities, we have labeled the nodes based on the highest activation at the 10 convolutional kernels in the first layer of DCNN. In Fig. 4 in order to provide visibility we have considered a sub-set of 200 nodes with at least 3 edges in BlogCatalog data. The nodes are colored based on the community learned by DCNN. It can be seen that DCNN can learn community embedding effectively. For example, we can see that blue and brown nodes are connected.

Lastly, we compare the output label vector from a neuro-fuzzy classifier with that of a feedback controller in Fig. 5. We can see that the former tends to classify all nodes into the same class, however the feedback controller is able to detect the positive class labels in the sparse network.

Conclusion

In this paper, we proposed a novel fuzzy convolutional deep walk for multi-label classification of nodes in social networks. Our proposed method outperforms baselines by up-to 15% in F-measure. In order to learn communities of node embeddings effectively, we consider deep convolutional neural networks. Furthermore, we capture the sequence of connected nodes in the network by using a fuzzy feedback controller. Deep learning uses contrastive divergence to heuristically approximate the weights of the network, which provides higher accuracy than baselines. The method is computationally fast and requires much lower complexity than baselines. Lastly, we are able to visualize the communities as highly activated nodes at each convolutional kernel in the deep model.

References

Cambria, E.; Wang, H.; and White, B. 2014. Guest editorial: Big social data analysis. *Knowledge-Based Systems* 69:1–2.

Cao, S.; Lu, W.; and Xu, Q. 2015. GraRep: Learning graph representations with global structural information. In *CIKM*, 891–900.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *AAAI*, 1145–1152.

Chang, S.; Han, W.; Tang, J.; Qi, G.; Aggarwal, C. C.; and Huang, T. S. 2015. Heterogeneous network embedding via deep architectures. In *SIGKDD*, 119–128.

Chen, G.; Ye, D.; Cambria, E.; Chen, J.; and Xing, Z. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *IJCNN*.

Cherman, E. A.; Monard, M. C.; and Metz, J. 2011. Multi-label problem transformation methods: a case study. *CLEI Electronic Journal* 14(1):1–10.

Fang, H.; Wu, F.; Zhao, Z.; Duan, X.; Zhuang, Y.; and Ester, M. 2016. Community-based question answering via heterogeneous social network learning. In *AAAI*, 122–128.

Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *SIGKDD*, 855–864. ACM.

Majumder, N.; Poria, S.; Gelbukh, A.; and Cambria, E. 2017. Deep learning based document modeling for personality detection from text. *IEEE Intelligent Systems* 32(2).

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2014–2023.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*, 701–710.

Poria, S.; Chaturvedi, I.; Cambria, E.; and Hussain, A. 2016. Convolutional MKL based multimodal emotion recognition and sentiment analysis. In *ICDM*, 439–448.

Tang, L., and Liu, H. 2009. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 1107–1116.

Tang, L., and Liu, H. 2011. Leveraging social media networks for classification. *Data Min. Knowl. Discov.* 23(3):447–478.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *SIGKDD*, 1225–1234.

Wang, H. O.; Tanaka, K.; and Griffin, M. F. 1996. An approach to fuzzy control of nonlinear systems: stability and design issues. *IEEE Transactions on Fuzzy Systems* 4(1):14–23.

Xing, F.; Cambria, E.; and Zou, X. 2017. Predicting evolving chaotic time series with fuzzy neural networks. In *IJCNN*.

Zachary, W. W. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 452–473.

Zheng, V.; Cavallari, S.; Cai, H.; Chang, K.; and Cambria, E. 2017. From node embedding to community embedding. https://arxiv.org/abs/1610.09950.