CS230

Graph Convolutional Networks for Friend Recommendation

Albert Pun
Department of Computer Science
Stanford University
apun@stanford.edu

Abstract

As more ethical concerns emerge about the use of user data by social networks, it is ever more important to respect user's privacy. However, many recommendations algorithms for ads and friends often exploit user data. In this study, we were able to do link prediction on a social graph with only unsupervised features between nodes. We passed these features into a graph convolutional neural network to get node embeddings and inputted in pairs of node embeddings to a full connected network to receive link predictions. We were able to achieve a competitive mAP@10 of 67% without using any user data other than their current friends.

1 Introduction

As social networks like Facebook and Twitter have grown at an exponential rate in the last several years, it has become ever more important for these platforms to help users find the friends they know in real life. To do this, these social networks need to understand which users are most likely to be connected to another user. Often, recommendation algorithms usually use the user's attributes and other links in the entire social graph to make an accurate prediction. However, in our research, we want to find out if graph convolutional neural networks can perform accurate link prediction based solely on other existing links without any user attributes. The inputs to our network are the features derived from links for each node along with adjacency matrix that represents the connections between nodes. The adjacency matrix has some links removed, and the network must output a list of link predictions that it predicts exists in the graph.

With the growing concern of social networks abusing user data, this network can offer competitive link prediction without the need to read user data. This will become even more applicable as more countries like Europe and the U.S. passes more laws that restrict the use of user data.

2 Related work

Understanding social relationships in a social graph with recommendation algorithms has attracted significant attention in recent years [1, 2, 3]. All of these algorithms share a key assumptions that a user's new relationships are similar to the people around them.

With friend recommendations, there are two classical approaches: a topology-based approach and a content-based approach. With topology-based approaches, the models uses properties from the network structure and calculates similarities between nodes. The algorithm will then compare a target node with every other node and output the node with the highest similarity score. The two most well-known node-to-node similarities metrics is Jaccard [4] and SimRank [5]. There are other metrics,

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

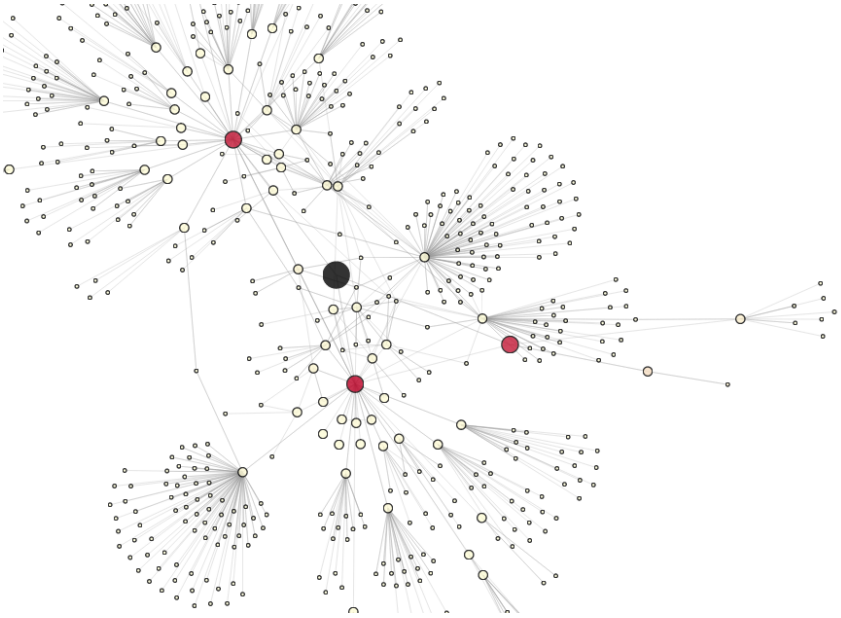

Figure 1: A training example, where the large black node is the target node. We only keep nodes with 5 degrees away from the target node to create a mini graph, so it can fit into memory

such as Zhao et al's P-Rank [6], which generalizes a version of structural similarities. Additionally Leicht et al [7] proposed another similarity metric that can be understood as a weighted count of the number of paths having possible length between two vertices. A content-based approach tries to recommend new nodes by looking at the current nodes linked to the target node. The most popular content-based approach is collaborative filtering, which is a method of making automatic predictions about the interests of a user by collecting preferences from many users connected to the target node.

There are many different types of social network recommendation applications that vary in regards to context, but are still relevant to look at. Lo et al [8] developed a topology-based model to predict strengths of relationships by understanding real messages between users. Armentano et al [9] was able to develop an unsupervised model using Twitter's data to identify users who were a trusted source by other users. Lastly, Yang et at [10] created a method to recommend friend requests that results in the highest acceptance rates.

3 Dataset and Features

The data is from Facebook's recruiting challenge on Kaggle [11] and contains a directed graph of 1.86M nodes and 9.43M edges. Each directed edge represents the source user following the destination user. The dataset only contains a single giant graph, but we need to split it into train, test, and validation. We decided to use 10% of the nodes for validation, 10% for test, and the last 80% for training.

Due to the large size of the network and the inability to fit all the data into memory, we had to come up with a way to do create training examples that fit in memory while still being representative of the original data. To do this, we created a mini-graph for each training example by sampling a target node from the entire graph and only keeping nodes within 5 degrees away from each target node to create a mini graph. For each training example, all link predictions were done relative to the target node. Figure 1 shows an example of a mini-graph with the target node.

To create the supervised problem, we removed a target node's link with 20% probability. We have the model try to predict the links that were removed.

To extract the initial features features that are inputted into the GCN, we used unsupervised features that were shown to provide rich information about its neighbors. We don't have any info about the

$$
\begin{aligned}
CN(u,v) &= \mid \Gamma(u) \cap \Gamma(v) \mid \\
JC(u,v) &= \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \\
AA(u,v) &= \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{log(|\Gamma(w)|)} \\
RA(u,v) &= \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(w)|} \\
PA(u,v) &= \mid \Gamma(u) \mid \times \mid \Gamma(v) \mid \\
AR(u,v) &= \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \\
ND(u,v) &= \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{|\Gamma(u)| \times |\Gamma(v)|}} \\
TN(u,v) &= |\Gamma(u) \cup \Gamma(v)| \\
UD &= \mid \Gamma(u) \mid \\
VD &= \mid \Gamma(v) \mid \\
SC(u,v) &= \begin{cases} 1 & \text{if u and v belong to the same community} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Figure 2: We used unsupervised features

actual node, so we use unsupervised methods based on Kolja Esders's work in [12]. The following features will be included in our prediction task: Common Neighbors (CN),Jaccard Coefficient (JC), Adamic-Adar index (AA), Resource Allocation (RA), Preferential Attachment (PA), Adjusted-Rand (AR) and Neighborhood Distance (ND) which are all local indices to which we will add the Total Neighbors (TN), Node Degree (UD et VD) and Same Community (SC) features. They are all defined in figure 2, where u is a non-target node, and v is a the target node.

4 Methods

We used a graph convolutional neural network to take the unsupervised features to output higher dimensional embeddings and using neighboring node's information. GCNs are in some ways are similar to a normal convolutional neural network. They perform similar operations where the model learns the features by inspecting neighboring nodes. The major conceptual difference between CNNs and GNNs is that CNNs are designed to work well on image data, where each pixel has neighboring pixels while GNNs are the generalized version of CNNs where the numbers of nodes connections vary.

The formula for each layer of the GCN is

$$X^{(l+1)} = ReLU(\widetilde{A}X^{l}W^{l}) \tag{1}$$

where each row of X represents the a node and its features. To create the original inputted feature vector for each non-target node, we used the equations from figure 3. $\widetilde{A}$ is a normalized version of the adjacency matrix with also the addition of the identity matrix.

By multiplying the $\widetilde{A}$ with X, the output node embeddings have information from its neighbors. It is important to note that every node is connect to itself in A so that the resulting embeddings have information about itself. Another vital detail is that each row of $\widetilde{A}$ is normalized so that nodes with large number of links do not grow too large and explode.

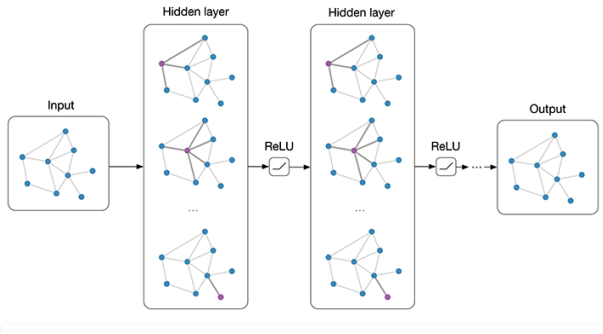

Figure 3: This figure illustrates how the purple node receives information from its neighbors in each layer. With multiple of these layers, the outputted embedding contain rich information from nodes many degrees away

After we use the GCN to find rich embeddings of each node, we then pass in pairs of node embeddings to a second full connected network that outputted the probability that the link existed in the original graph.

We used a cross entropy loss because the problem reduces down to a binary classification problem for each possible link in the graph. Due to the edges being directed, we needed to pass in each pair of nodes twice in reversed order.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i)) \tag{2}$$

5 Experiments/Results/Discussion

The evaluation metric for this competition is Mean Average Precision @ 10 as specified by the Kaggle challenge [11]

For example, if there are m missing edges from the target node in the graph, and you can predict up to 10 other nodes that the user is likely to follow. We can adapt the definition of average precision in information retrieval, where it is equal to

$$mAP@n = \sum_{i=1}^{n} \frac{P(k)}{min(m, 10)} \tag{3}$$

if the denominator is zero, the result is set zero. The numerator, P(k), is the precision at cut-off k in the item list. In other words, the ratio of number of users followed up to the position k over the number k, and P(k) equals 0 when k -th item is not followed upon recommendation. We chose n=10 because this was the way Facebook defined the evaluation metric for the Kaggle challenge.

We chose to use 4 layer GCN after trying 1 to 6 layers shown in Table 1. It is surprising to see that a GCN with just one layer is able to have decent performance, and that the performance increase for the addition of each layer caps out at around 4 layers, which is quite shallow when compared to CNNs.

We had to use a small mini batch size of 8 because our graph data took large amounts of memory despite our strategy of breaking it up as discussed in the Data and Features section.

GCN Layers	1	2	3	4	5	6
mAP@10	0.58	0.62	0.66	0.67	0.66	0.67

Table 1: We tried varying number of layers in our GCN network that produced the highest mAP@10

class	Predicted Yes	Predicted No
Actual Yes	76%	24%
Actual No	46%	54%

Table 2: The confusion matrix for our model

Total Edges	1-10	10-100	100+
mAP@10	0.55	0.69	0.71

Table 3: We bin each training example based on the number of edges a target nodes has.

For our model that received the input and outputted the link prediction, we chose to use a 3 fully connected layer model to do the classification because we wanted to train the network end to end in Tensorflow [13].

We also used L2 regularization along with dropout for all training to reduce overfitting. Our training accuracy was 72% while our test accuracy was 67%, which means that are model was still overfitting to some degree despite our regularization techniques.

Our results show that just by using unsupervised features we can score a 67% ap@10, which are competitive results to other methods that achieved 74% [1] even though we do not use any user attributes.

We found that the algorithm performed poorly when the target node had few connections as shown in Table 3. This is most likely due to the fact that there were less neighbors to learn from, making it harder

6 Conclusion/Future Work

Our research has shown that by just using unsupervised features to do link prediction on social graphs using GCNs, we can achieve competitive results to other studies that utilize user attributes. The message passing in GCNs has shown to create rich embeddings that can be used by a FCN to do link prediction between pairs of nodes.

In the future, other teams could investigate how to produce link predictions over an entire graph of each node without needing to individually input pairs to the fully connected layer for a specific target node. This would drastically increase the speed of the network during inference time.

Future work can also be done on using different features for the initial input to the GCN.

Teams with larger computing power can also try to input larger training examples to see if it improves performance. In our case, we could only use nodes 5 degrees away from target nodes because it took too much memory. By distributing the data over hundreds of GPUs, it may even be possible to run the whole 10m node graph in one pass to get better results.

7 Contributions

Albert Pun worked independently on this project.

8 Code

https://github.com/AlbertPun/cs230project

References

[1] Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Recommendations in signed social networks. In Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 31–40.

[2] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2013. Exploiting local and global social context for recommendation.. In IJCAI, Vol. 13. 2712–2718.

[3] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. 2017. Social collaborative filtering by trust. IEEE transactions on pattern analysis and machine intelligence 39, 8 (2017), 1633–1647.

[4] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1983.

[5] G. Jeh and J. Widom, "SimRank: a measure of structural-context similarity," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2002, pp. 538–543.

[6] P. Zhao, J. Han, and Y. Sun, "P-Rank: a comprehensive structural similarity measure over information networks," in Proceedings of the 18th ACM conference on Information and knowledge management. ACM, 2009, pp. 553–562.

[7] E. Leicht, P. Holme, and M. E. Newman, "Vertex similarity in networks," SIAM Journal on Numerical Analysis, vol. 45, no. 2, pp. 890– 904, 2006

[8] S. Lo and C. Lin, "WMR–a graph-based algorithm for friend recommendation," in Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society, 2006, pp. 121–128.

[9] M. G. Armentano, D. Godoy, and A. Amandi, "Topology-based recommendation of users in micro-blogging communities," Journal of Computer Science and Technology, vol. 27, no. 3, pp. 624–634, 2012.

[10] D.-N. Yang, H.-J. Hung, W.-C. Lee, and W. Chen, "Maximizing acceptance probability for active friending in online social networks," in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013, pp. 713–721.

[11] Facebook, "Facebook Recruiting Competition," Kaggle, https://www.kaggle.com/c/FacebookRecruiting/data

[12] Kolja Esders, (2015). Link Prediction in Large-scale Complex Networks. Bachelor's Thesis at the Karlsruhe Institute of Technology

[13] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/