COMP90051 Statistical Machine Learning

Project 1: In-class Link Prediction, Semester 2 2018 Report

Yun Chen and Geoffrey Ka-Hoi Law
{yunc4, glaw}@student.unimelb.edu.au

Abstract

In the project, we implement a machine learning system which predicts the likelihood of existence of an edge between two nodes in a graph. In this report, we describe our approach which uses a supervised machine learning method for classifications. Furthermore, we explain the model used and the features selected, we also evaluate different models in order to choose the best one for *Kaggle* submissions. Finally, we address some issues and our important findings.

1 Introduction

Link prediction is a binary classification problem which to predict whether an edge between two nodes in a graph is real or imaginary. By using different supervised classifiers [5], we will be able extracted features from the network and train the models to tackle such problem. Furthermore, to improve our results, an attempt was made to learn network structure and to predict the existence of connection by using embedding, though not successful.

2 Dataset

The dataset is a subgraph of entire *Twitter social network* which consists of approximately 4,800,000 nodes and 24,000,000 edges, where each node represents a Twitter user and each edge represents a friendship between two nodes.

3 Supervised Learning Approach

We implemented our supervised learning system in Python, using *scikit-learn* [7] and TensorFlow[1] (with keras API) machine learning package for data pre-processing, model training and model evaluations. In addition, by many trail and errors have been done previously, we come up an efficient graph storing strategy which is using *networkx* [4] as the graph representation for storing the entire network.

Since the size of training data is very large, we randomly select a subset (around 6,000,000 entries) of training data which has smaller size for model training. We evaluate on many different classifiers in order to choose the best one.

3.1 Feature Selections

By inspiring from [6] [9], we choose six candidate heuristics to be selected as features; common neighbors, Jaccard coefficient, preferential attachment, Adamic-Adar index, resource allocation index, and Katz centrality.

These features can be categorized into first-order, second-order and high-order features [9], as shown in Table 1. The first-order feature considers 1-hop neighbors of a pair of nodes while the second-order feature considers 2-hop, and the high-order feature requires knowing the entire network. The high-order features provide significant information in link predictions. However, the calculation of high-order features e.g. centrality, PageRank etc. are time-consuming when the network is large. Therefore, we deprecate high-order features.

In our final approach, by feature selection functions in *scikit-learn* package, as well as throughout many tests, we decide to include Jaccard coefficient and resource allocation index only. Before using the features in the classifier, we normalize their values to have zero mean and unit variance.

Name	Formula	Order
common neighbors	$\mid \Gamma(x) \cap \Gamma(y) \mid$	first
Jaccard coefficient	$\frac{\mid \Gamma(x) \cap \Gamma(y) \mid}{\mid \Gamma(x) \cup \Gamma(y) \mid}$	first
preferential attachment	$\mid \Gamma(x) \mid \cdot \mid \Gamma(y) \mid$	first
Adamic-Adar index	$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log \mid \Gamma(z) \mid}$	second
resource allocation index	$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\mid \Gamma(z) \mid}$	second
Katz centrality	$\sum_{l=1}^{\infty} \beta^l \mid \text{walks}^{\langle l \rangle}(x.y) \mid$	high

Table 1: Heuristics for Link Prediction

3.2 Model Evaluations

We test various machine learning models including SVM (RBF and Liner kernal), K-Nearest Neighbors (KNN), Logistic Classification, Bagging, Multilayer Perceptron (MLP) and Neural Network (2-64-64-1 layers, nadam optimizer and binary cross-entropy loss function). SVM, KNN, Log, Bagging and MLP were using the same settings as in [5], as coefficient = 8 for SVM and k = 12 for KNN to be optimum. The scores for each model are shown in Table 2. The result turns out SVM (RBF Kernel) is the suitable model for link predictions in supervised learning, with NN closely following behind. The best AUC we have got is around 0.85 by using supervised learning.

Name	AUC
SVM (RBF Kernel)	0.8497
SVM (Linear Kernel)	0.8487
K-Nearest Neighbors	0.8473
Logistic Classification	0.8179
Bagging	0.7903
Multilayer Perceptron	0.8483
Neural Network (TensorFlow)	0.8473

Table 2: Models for Link Prediction; features are Jaccard and resource allocation as features; tested with test-public.txt and the AUC scores are obtained from *Kaggle*

In this case, NN works relatively well, but do not surpass SVM. The reason behind that is, the link prediction is a simple binary classification problem, with only 2 input features and 2 labels (True, False) in our case. For these kind of tasks, Neural Networks are unnecessarily complicated - their neurons and perceptions are not generating useful extra features. In essence, NN are much more suitable for more complicated works.

SVM (RBF Kernel) performs the best in our evaluations. As [5] concluded, link predictions require to bias the model on predicting more positive links than negative links, which can be significantly achieved in the classifier that are norm-based such as SVM and KNN by assigning higher cost to the misclassification of positive links.

4 Further Attempts and Issues

In attempts to improve our AUC over 0.85, we decide to try out embedding, including both node2vec [3] and DeepWalk [8]. Embedding is a technique to convert the network nodes into lower dimensions, while preserving the network structure [2]. However, since the network is extraordinarily large (approximately 24,000,000 edges), full scale node2vec embedding was not possible due to memory and computational constrains. We try to limit embeddings to only 1-hop sub graph of testing nodes, but result is not successful, with only 0.5 AUC. Due to similar circumstances, the attempts for [8] was not successful. DeepWalk [8] is indeed a lot more feasible than node2vec [3], but we still lack the resources to complete it.

In fact, during this project, we felt the limitation of computing resources. The sampling of

3,000,000 positive entries takes over a week to complete. The feature extractions on larger nodes was extremely slow, which later on we have to limit the sampling size on such nodes. The sampling of negative entries was fast, due to simple network structure. To push these 6,000,000 entries into different models, we almost used up 25GB of RAM. Fortunately, the training and prediction speed are fast.

In a similar manner, when we tried a framework proposed by Zhang et al. [9] which uses Deep Graph Convolutional Neural Network (DGCNN) [10] as classifier, we ran into problems such as MATLAB code was consuming way to much memory than we can afford, and subgraph extraction is overly time-consuming. We reported the problems to the authors and they kindly replied for appreciations and promised for future improvements.

5 Conclusion

In this project, several different supervised learning models were explored and implemented to handle link prediction problems. We have chosen the features that suitable for the problem and have evaluated different classifiers in order to choose the best model. For feature selections, we have referred to some idea from many relevant literature such as [5] and [9]. In attempts of using embeddings approach to further improve our result, many problems were encountered such as lack of memory and computational power, so these further attempts were not successful. In conclusion, by using supervised learning, SVM (RBF Kernel) as model, Jaccard and resource allocation as features, we obtained our best AUC is approximately 0.85 in *Kaggle*.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] P. Cui, X. Wang, J. Pei, and W. Zhu. A survey on network embedding. *CoRR*, abs/1711.08752, 2017.

[3] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[4] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[5] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. 2006.

[6] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014.

[9] M. Zhang and Y. Chen. Link prediction based on graph neural networks. *CoRR*, abs/1802.09691, 2018.

[10] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, pages 4438–4445, 2018.