COMP90051 Statistical Machine Learning
Link Prediction for Graph Networks

Alex González, Yuqing Xiao, Yee Hean Chuah

1. Introduction

Social media is increasingly becoming an indispensable aspect of our social lives. Whether it be building personal or professional relationships, it might be of interest to recommend relevant users to each other. Therefore, an automated friend recommender system can be useful whereby identifying potential links are beneficial. In the following sections, we examined the suitability of several models in building an edge classification model, namely Naïve Bayes, Multilayer Perceptron and Logistic Regression. Due to its ability to self-learn feature representations, we expect the Multilayer Perceptron to outperform both Naïve Bayes and Linear Regression. To test our hypothesis, we introduced multiple methods for comparison throughout the process.

2. Description of Dataset

There is a total of 20,000 nodes in the dataset, with approximately 24 million edges between them. Some of them contain duplicates and create self-loops. However, these only comprises 0.2% of our total instances, which is negligible. Thus, the faulty instances were removed directly from the dataset.

3. Sampling

Given the large graph, the first challenge is to derive a representative sample for the sake of improving computational cost and time. A subnetwork is representative if it has similar structural properties as the full network. However, as the training dataset was pre-crawled, we do not possess any prior knowledge on the network topology, presenting an additional challenge. We used a variant of Random Walk and Jump algorithm for sampling edges because of its good performance as shown by [3]. An initial seed was picked at random from the training dataset, and the next node was selected from its sinks. This process is repeated until we obtain a trail of sampled edges. With a probability of 0.1, our tracer would "hop" to another node, which helps to avoid getting stuck in a local neighbourhood.

4. Feature engineering

Regularization

Though L1 can be more stable, L1 has the function of choosing features which is not very useful in our project as we don't have many features. When Gaussian Markov condition is met, it can be proved that the parameters obtained by using the L2 loss function are unbiased and effective. In the fitting process, it is generally inclined to make the weights as small as possible. Because it is generally believed that models with small parameter values are simpler, can adapt to different data sets, and avoid over-fitting to a certain extent.

Standardization

We tried two standardization, first is Min-Max, second is Z-score. Z-score will consider the mean and variance of the whole column when Min-Max will be influenced by very big maximum or very small minimum, so we use it finally.

Why we use them

First, normalization/standardization can avoid some unnecessary numerical problems. Because the nonlinear interval of tanh is approximately [-1.7, 1.7], for the neuron to be effective, the magnitude of w1x1 + w2x2 + b in tanh should be around 1. Second, we hope that the input of the function and the initialization of neurons can be within a reasonable range. Third, if the gradient is very large, the learning rate must be very small. Therefore, it is better to normalize the data directly, so that the learning rate does not need to be based on the data to adjust the range.

Given the dataset is represented by sources mapped to its respective sinks, we first convert them into source and sink pairs. To enable supervised learning, we need positive and negative samples. Since all the edges in our training set are real, creation of fake edges is required. We assume that each edge is a random variable and that the distribution of fake edges is different from that of original data. The fake edges are generated based on a variant of Random Walk algorithm. Starting from a random node, with p =0.5, we uniformly select one of its sinks to "walk" to. The algorithm has a 50% chance of stopping its walk, and an edge is generated between the seed and final node. This allows creation of more realistic fake edges with varying extensions.

An interesting observation here is the directed graph. Unlike the concept of 'friends' in Facebook, edge direction matters in Twitter. Therefore, it is necessary to discern between followings and followers. Table 1 shows our features.

Feature	Description
1)Sink_Followers	How many people is following the sink user?

2)Common_Followings	How many people is following both sink and source?
3)Source_Followings	How many people is following the source user?
4)Source_Followers	How many people the source user is following.
5)Distance	The shortest distance between sink and source.
6)Inv_Indirect_Followings	The inverse of transitional followings.
7)Com_Followings_Ratio	Common followings / number of followings.
8)Followers_Ratio	Common followers / number of followers.
9)Indirect_Followings	Transitional followings. (e.g. consider a following chain A-B-C, A indirectly follows C
10)Friends_Cycle_Num	In source's following and those followings' following, how many people is following sink.
11)Source_Friends_Num	In source's following and those followings' following, how many people is following source.
12)Source_Followers_ToSink	Common followers. The number of followers of the source that also are followers of the sink.

Table 1 Features description

Features 1-4 represents numbers of following and followers that sink and source have or share. These features will indicate if the node is an active user who like to follow others or are likely to be followed by others (for example, a celebrity). Common followings indicate how similar sink's and source's followings are—how many people have interest in both, and in some way, it tells us how similar sink's and source's friend circle is. Feature 'Distance' represent how close sink and source are in the social network graph. 10 and 11 is based on searching from source as the root and depth as 2. 10 is the number of how many source's following is interested in sink. When paired with 1st feature it shows how big part of sink's followers are friends instead of fans. 11 tells us how many of source's followers are "friends" instead of "fans".

In addition, it was stated testing set has a 50-50 split between real and fake edges. A fundamental assumption in machine learning is that the training and testing dataset is drawn from the same population, hence should have the same distribution. Therefore, we ensured the number of fake and real edges remain equal throughout our training process. Closer inspection suggests that the number of followings lie over a greater range than the number of followers. This may result in the "following" feature dominating the "followers". A min-max normalization was performed to rescale the feature over the range [0,1], ensuring all features are represented equally during training. This also helps speed up learning and leads to faster convergence in neural networks.

5. Proposed Learners

The following model were implemented:

- **Naïve Bayes**: Initially performed as baseline.
- **Ridge Regression**: Ridge linear regression.
- **Logistic Regression:** Logistic regression model with L2 regularization and 'liblinear' as the solver
- **Neural Network:** We use a simple neural network with two hidden layers. Loss function used is binary cross entropy with 50 epochs trainings.

These 4 learners are selected for several reasons: The Naïve Bayes uses a simple learning rule by modelling P(x,y) and obtain conditional distribution using the Bayes' Rule. The underlying conditional independence assumption makes it robust, and scales linearly with number of instances. Further, we use Logistic Regression, which models P(y|x) directly. Mainly, it is used to observe the performance disparity between the generative and discriminative models. Finally, Neural Network was selected because of its exceptional representational learning, and to serve as a "gold standard" benchmark for the simpler models. We proceed to train our model with the 12 generated features. The performance of the proposed learners is shown in Table 2. Also, in Figure 1 it is presented the Loss function analysis for the Neural Network model. These results are discussed in Section 6 and 7.

Model	AUC on training dataset	AUC on Kaggle test
Naïve Bayes	1.00	0.5935
Ridge Regression	0.86	0.7012
Logistic Regression	0.93	0.7432
Neural Network	1.00(0.88 after adjust)	0.7953

Table 1 evaluation table of models

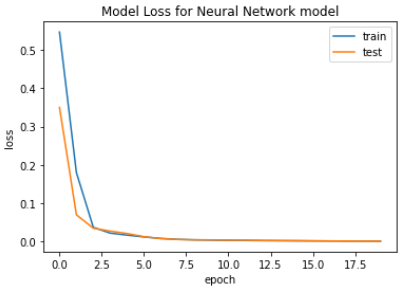

Figure 1 Learning Curve of NN

6. Discussion and Evaluation

Naïve Bayes is the weakest performing model here. This is because our generated features are highly dependent. For example, two nodes with high number of common followers tend to also have a lot of common followings and short distances between them. This is consistent with the findings of [1], who describes these as "community" nodes. In real life, it would be akin to two best friends with highly overlapping social circles. Thus, the conditional independence assumption does not hold here. Another reason is that Naïve Bayes produces a higher asymptotic error when training data is large. [2]

Logistic regression achieved an AUC of 0.74, which is higher than Naïve Bayes. Unlike Naïve Bayes, logistic regression has no implicit feature independence assumption, therefore it can capture the relationship between features during training. This makes it ideal for our highly correlated features. Neural Network model yields an AUC of 0.79, which is the best score. This model learns through backpropagation algorithm where features are automatically learnt based on the updated weights, which explains its high performance. However, the heuristic approach Neural Network adopts meant that the results generated are often challenging to explain, compared to the classical machine learning algorithms.

To allow further experimentation, we created additional training datasets comprising various subset of feature combinations, and real / fake edges. Our experimentation indicates distance is a key metric when it comes to identifying potential links between nodes. This is sensible because in the real-world setting, a person is more likely to know another who is 2 degrees away from them (ie. A friend of a friend) as compared to 3 degrees. This is further evidenced from our prediction result, where we observe a strong positive correlation between a positive prediction and the inverse distance. Shorter distances are more likely to be predicted as a 'True' edge.

7. Error Analysis

Both of our models show substantial improvement beyond the baseline; however, the current level of performance suggests that it is still far from feasible in real world applications. As Neural Network is the best performer, the error analysis on this model is our focus. As it is shown in Table 2, there is a significant disparity between training and testing AUC. The Loss function performance shown in Figure 1 for Neural Network, suggest that the underperformance is not due to overfitting. Manual inspection indicates the decrease was due to the distance metric. In our training set, the distance is always 1 for every real edge, this was not always true in testing set, thus there may be overfitting present. We tried to implement a 2nd shortest distance metric in response, but another problem arises as some nodes are only accessible through a single pathway, thus returning infinity values.

Another explanation for the underperformance could be related to the sampling process. As discussed above, it was performed with Random Walk variants. This can be problematic due to degree bias introduced. This might result in a disconnected subsample from the training graph, which is already a sampled graph itself. The resulting low-quality sample may be the key reason our model performance plateaued despite repeated tweaks. To gain better insights, it is recommended to use statistical metrics such as D-statistic to measure the representativeness of sampled data. In addition, our model performance in the private leader-board was higher than in public which has a smaller dataset. This holds true for most groups too. This further flag our sample did not encode the topology of the neighbourhood well enough.

8. Conclusions and Future Directions

Our results on this graph classification task shows that Multilayer Perceptron outperforms both of Naive Bayes and Logistic Regression. However, it should be noted choosing the right model is not the only factor to achieve better predictions. Sampling and feature generation methods are equally important. For future work, consider exploring different sampling methods such as Forest Fire and Metropolis Hastings to get higher quality samples. Also, other models such as SVM could be useful for comparison against the former models.

References

[1] S. Pandhre, M. Gupta and V. Balasubramanian, "Community-based Outlier Detection for Edge-attributed Graphs", *Arxiv.org*, 2020. [Online]. Available: https://arxiv.org/pdf/1612.09435.pdf. [Accessed: 13- Sep- 2020].

[2] Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and Naive bayes. In Advances in neural information processing systems (pp. 841- 848).

[3]J. Leskovec and C. Faloutsos, *Cs.stanford.edu*, 2016. [Online]. Available: https://cs.stanford.edu/people/jure/pubs/sampling-kdd06.pdf. [Accessed: 18- Sep- 2020].

[4]N. Benchettara, R. Kanawati and C. Rouveirol, "Supervised Machine Learning Applied to Link Prediction in Bipartite Social Networks - IEEE Conference Publication", Ieeexplore.ieee.org, 2010. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5562752.