Assignment 1 report

Problem description

Pairwise relation is very common in real life and can be reflected as mutual following relationships among friends in social networks. In this assignment, we are going to design a model that can discriminate whether a user has a relationship with another or predict whether a user will have a relationship with another in the future.

Related works

In this assignment, we mainly referenced 1). The survey report published by Linyuan Lü about the AUC evaluation criteria and similarity features [1] and 2.) An analysis report of link predictions based on Weibo published by Ying-bin FU [2]. The considerable and deep insights provided us with a lot of ideas to help with our feature extraction.

Sample Selection

Issue Faced

In the beginning, we started processing the raw data to generate an edge set which contained all the edges in the network graph. We randomly chose real edges from the set and created fake edges from any two existing nodes in the graph. After feature extraction, we trained our model by Logic Regression. We have got over 90% accuracy on the validation data set. However, when we submitted our predictions of the test set to Kaggle, we only achieved AUC 45.5%.

Troubleshooting

The extreme gap of accuracy score between validation data and test data raised our doubts about the reliability of our training data. After checking the predictions of test data, we were aware that most of the edges in the test set are classified as positive, around 1700/2000. The numbers of positive edges were supposed to be 1000 in the test set, and it indicated our model preferred to predict edges as positive. We realised we might generate negative samples in the wrong way and this caused troubles for our model to identify and predict fake edges. After analysing edges in the test, we figured out a fact. In the graph, we have 20000 seed nodes which are captured, a lot of sink nodes which are also captured, yet apart from those, there are tons of sink nodes which are not captured. The real edges are intuitively identified as (seed, captured node) while the remaining edges fall into two types. The first type is a fake edge which has the edge structure as (captured node, uncaptured node). And the second type of edges is sourced with an uncaptured node, which in this case should be classified as an uncertain edge instead of a fake edge. The problem here was we misclassified the approach to structure a fake edge, therefore our negative edge sampling was wrong. After making adjustments and recreating the training set, our model got AUC 71.1% on a test set with the same features.

Discussion

After feature extraction and model training, we also found that our models did not perform well on predicting some special edges. In the future, we can use over-sampling or under-sampling methods to adjust the proportion of special edges accordingly. For example, because the number of uncaptured nodes is far more than captured nodes, when we randomly choose negative samples, the number of edges from captured node to the uncaptured node will be extremely larger than edges from captured node to captured node. This implies our model has an insufficient ability on judging edges with both captured nodes on sides.

Feature

Feature engineering

After we did research, we found that the current link prediction algorithms mostly use two types of features in general. One is based on the neighbours of the node, and the other is based on the path from one node to the other. In this section, we will discuss and analyse the features that we used, most of which are included in Linyuan Lü’s report [1] and the result of using them.

- Common Neighbors: The first feature that came to our mind is the number of common neighbours of two users, two users are more likely to establish a link if they have many common neighbours.
- Preferential Attachment Score: There is a phenomenon in social networks that people with more friends are more likely to become friends, that is to say, "elites" are more likely to form relationships. This can be seen as the *Matthew effect*. A pair of vertices with larger chain eigenvalues is more likely to form an edge, which is represented by the product of the neighbours of two vertices.
- Adamic-Adar Index: This feature can be considered as the improvement of common neighbours. When we calculate the number of the same neighbours at two vertices, the "importance" of each neighbour is different. We believe that the less the number of connected nodes of a neighbour, the more prominent its "middleman" property.

In addition to the above two features that we have come up with to represent the relationship between users, we have also referred to many features in the study of other researchers[2], but we can not introduce them one by one because of the limited space.

Pros and cons

When we initially only used the features with common neighbours, we only got 70% accuracy on Kaggle. When we gradually added features such as *Adamic-Adar* and *Jaccard* that emphasize the “middleman”, our accuracy rate slowly rose to more than 75%, which shows that the middleman plays an important role among users in the *Twitter* network.

In Lü's report [1], we can find that even a feature that performs well on one data set may not perform well on another data set, and this is also reflected in our model. During processing our attribute extraction, we spent a lot of time extracting a subgraph feature based on an $O(n^2)$ algorithm to describe the clustering coefficient of the neighbours of the two vertices. In FU’s report [2], the Gini index of this feature is among the top. But when we trained the models with this feature, the results did not rise but fall. We assume that this may be because our data set was collected several years ago when *Twitter* users were relatively few, an account may provide a variety of information, therefore, the degree of clustering between neighbours cannot bring much information.

Discussion

As mentioned before, there are two kinds of features widely used to achieve link prediction, but because our training set is too large with a total of 400,000 edges, we exacted neighbour-based features mostly. If more time is allowed, we might try extracting some path-based features for training to achieve better results. It is also worth mentioning that in our continuous attempts to search effective features, we found that a simple binary feature *Opposite Direction Friend* also helped in a way. This reveals that if a user follows the other user, then the other is likely to follow the user back as well, which also tells us that the amount of information carried by the feature is not strongly related to its computational difficulty.

Model

Learner selection

We came up with the three supervised machine learning models to solve this link prediction problem, Logistic Regression (LR), Neural Networks (NN) and Support Vector Machine (SVM). After building the three models, we applied the same training set among them to predict our validation set and compare the outcome results of theirs. The SVM produced the least accuracy on the prediction at around 62%, whereas the other two all reached the accuracy above 75%. We, therefore, chose SVM as our baseline model, and our focuses were shifted to optimise and tune LR and NN models.

Fine Tuning in LR

As a result of predictions made by LR model using two different regularisation methods, the accuracy under L1 regularisation was around 84% while the score was only 75% with L2 regularisation. It raised our interests of why L1 did a better job than L2 with nearly 10% higher accuracy. We analysed this case and assumed the reason might be L1 has better robustness than L2. Most features extracted are based on the neighbours of nodes, and a few nodes have an extremely huge number of neighbour nodes like over 500000 or an extremely small number like 0. L1 has a better capability to tolerate these extreme data. What is more, L1 can produce a sparse model which can help filter features. Since we have over 29 features, some of them may not be meaningful enough to influence the accuracy of prediction and cause overfitting problems. L1 regression can punish the severe penalty in the case and reduce the weight of unimportant or noisy features to 0. But there is a drawback of L1. It costs more time for computing models. We can finish model training within several seconds with L2 while it takes over ten minutes with L1. For the solver parameter, because only saga and liblinear support L1 and saga is more appropriate for big training data sets, thus we chose saga and used cross-validation to determine the best parameter C.

Final approach

We experimented with different tuning combinations for the NN model as well, with the parameters of numbers of the hidden layers, hidden nodes for each layer, the maximum iterations and the active functions. After tuning both LR and NN, we ran the models to make probability predictions. The accuracy of LR on the validation set achieved 93%, while the accuracy of NN was around 90%. Hence, we chose this LR model as our final submission. Our NN model may not be best optimised to achieve the best accuracy because we do not have enough hands-on experience on configuring it. We analysed the reasons why LR shows high capability in this case based on our current knowledge. In our features, there are eight features computed based on the numbers of neighbors over the nodes (e.g. Hub Depressed Index [1] and Salton Index [1]). These features are not independent (e.g. changing the value of common_neighbors will influence the value of all these features) and may provide redundant information. The regularization of L1 helps reduce the impact of multicollinearity because it can filter redundant features. As regarding the NN model, we noted that we did not add a “dropout” layer in our NN model to do regularization. This may be a reason why our NN model had worse performance than LR. In terms of future work, we shall do more experiments with diverse features for example, the path-based features rather than the neighborhood-based features which are mostly used in our project. We believe those new features may improve our model performance in a way.

Reference

[1] L. Lü and T. Zhou, "Link prediction in complex networks: A survey", Physica A: Statistical Mechanics and its Applications, vol. 390, no. 6, pp. 1150-1170, 2011. Available: 10.1016/j.physa.2010.11.027 [Accessed 17 September 2020].

[2]Y. FU and Y. CHEN, "Relationship Analysis of Microblogging User with Link Prediction", Computer Science, 2014. [Accessed 17 September 2020].