COMP90051 Statistical Machine Learning

Na Chang (nchang1 - 858604)
Junhan Liu(zayhan - 878637)
Zepeng Dan(zdan - 933678)

1. Problem Context

In social network relationship graphs, users are modelled as nodes and the relationships between them as edges. This project aims to predict the edge between two nodes, utilising the outbound neighbours.

2. Dataset and Evaluation Method

2.1. Dataset Description

The training dataset is a subgraph of Twitter social network. It is a directed graph, and it only contains the outbound neighbours of each node. The test set has 2,000 edges, 50% of which are real edges.

2.2. Evaluation Method

The project uses AUC as the main evaluation matrix. And a random model is developed to provide baseline for performance measurement.

3. Feature Set

3.1. Overview

The training edges are extracted based on below rules:

- Due to the intractable size of training data, the graph is pruned to only keep the relevant nodes in the testing dataset. And the edges between the relevant nodes and their first level neighbour are extracted as training data.
- Considering the proportion of positive – negative items in the testing data, 50% of the training data are positive edges, and the other negative cases, which are extracted using random index in the user list.

3.2. Calculated Index

Six similarity indexes are calculated separately based on the outbound and inbound sets between a pair of nodes. Some notations are illustrated here. $\Gamma(X)$, $\Gamma(Y)$ denotes the set of neighbours around node X and Y. C_x, C_y denotes the number of neighbours around X and Y, which could also be represented as $|\Gamma(X)|$ and $|\Gamma(Y)|$.

Similarity Name	Formula
Node degree	$\|\Gamma(X)\|,\ \|\Gamma(Y)\|$
Sum/difference between node degree	$\|\Gamma(X)\| + \|\Gamma(Y)\|$ $\|\Gamma(X)\| - \|\Gamma(Y)\|$
Common neighbour	$\|\Gamma(X)\| \cap \|\Gamma(Y)\|$
Jaccard Index	$\frac{\|\Gamma(X)\cap\Gamma(Y)\|}{\|\Gamma(X)\cup\Gamma(Y)\|}$
Salton Index	$\frac{\|\Gamma(X)\cap\Gamma(Y)\|}{\sqrt{C_x * C_y}}$
Priority Link	$\|\Gamma(X)\| * \|\Gamma(Y)\|$
HPI (Hub Promoted Index)	$\frac{\|\Gamma(X)\cap\Gamma(Y)\|}{min(C_x, C_y)}$
HDI（Hub Depressed Index）	$\frac{\|\Gamma(X)\cap\Gamma(Y)\|}{max(C_x, C_y)}$
Sorensen	$\frac{2 * \|\Gamma(X)\cap\Gamma(Y)\|}{C_x + C_y}$
Resource Allocation	$\sum_{z\in\Gamma(X)\cap\Gamma(Y)} \frac{1}{d_z}$

3.3. Feature Selection

Although twelve types of index are calculated for our project, not are the types of index are used as our features. The progress of feature selection is executed to refine the most valuable features. The valuableness of feature is presented as regularly and accurately separating positive and negative samples.

The following is our test result on feature selection. The selection is based on logistic regression with tolerance set to 0.0001. Three round tests are made and result are evaluated by AUC.

Feature Name	Exp1	Exp2	Exp3
Node degree(source node)	0.628	0.632	0.631
Node degree(target node)	0.563	0.567	0.563

Sum between node degree	0.632	0.631	0.634
difference between node degree	0.678	0.672	0.675
Common neighbour	0.674	0.670	0.668
Jaccard Index	0.832	0.831	**0.837**
Salton Index	**0.886**	0.882	0.883
Priority Link	0.611	0.612	0.616
HPI (Hub Promoted Index)	**0.898**	0.895	0.895
HDI（Hub Depressed Index）	0.846	**0.848**	0.844
Sorensen	**0.864**	0.858	0.864
Resource Allocation	0.664	0.661	0.669

The best results are shown in the table above with bold font. As indicated by the result, the first four features and priority link feature are deprecated because of its little contribution to classification problem. The Resource Allocation and Common Neighbour features are kept for their relatively better performance in later experiment.

3.4. Feature Normalization

Due to the difference among feature scales, we normalize all the input features before applying to our model. The method applied is min-max normalization.

$$X^{(i)} = \frac{X^{(i)} - X^{(i)}_{min}}{X^{(i)}_{max} - X^{(i)}_{min}}$$

where i denote the ith feature.

With normalization processing, all the features are mapped to the range of 0 to 1.

4. Model Evaluation

4.1. Hyperparameter discussion

The hyperparameters are searched basing on random search method[7]. This method perform relatively better than grid search method in our experiment. Empirical idea provide us with basic optimisation direction such as less units with more layers when training neural network could highlight the contribution of individual features. The data set is split into training set, develop set and test set in the ratio of 8:1:1 while guaranteeing the equal of positive and negative samples.

4.2. Baseline Model

A baseline model is developed, where the model will randomly assign 0 or 1.

Hyperparameter	AUC
N/A	0.43

4.3. Logistic Regression

A logistic regression model is developed

Setup	AUC
liblinear / l1 penalty	0.96
sag / l2 penalty	0.93
newton-cg / l2 penalty	0.93

4.4. Random Forest

A Random Forest model is developed.

Setup	AUC
entropy / min_split=4	0.968
entropy / min_split = 2	0.98
gini / min_split = 4	0.969

4.5. Neural Network

Neural Network model is developed with loss function set to binary cross entropy

Setup	AUC
hidden layer = 3 / unit = 20 / epoch = 50	0.96
hidden layer = 4 / unit = 20	0.95
hidden layer = 3 / unit = 30	0.93

5. Analysis and Improvement

5.1. Analysis

Dataset and Feature Analysis

Through studying the training dataset and features, below observations are made:

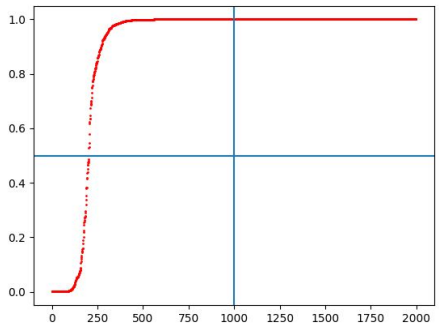
- Some accounts have a very large set of outgoing edges, reaching almost 800,000, while other accounts have no outgoing edges. Same observations apply to the incoming edges.
- Due to the large difference, the calculated features are very unbalanced in terms of scale.

Based on the above observations, the calculated features need to be adjusted to be on roughly the same scale. And the logarithmic approach is

considered, whose details will be stated in the next section.

Hyperparameter Analysis

When optimizing neural network model, more attention have been paid on overfitting problem. Although it shows we receive a fairly great AUC under the hyperparameter we set and the model, the practical result on the real test data is not ideal. As we draw the prediction curve with x axis be the order of sample and y be the prediction probability, the shape of the curve is far unlike with sigmoid function.



To handle the situation of this figure, two ways of overfitting optimization is put into practice. Dropout and penalty are applied to arbitrarily drop part of the units and lower the weight of some extremely strong feature.

Problem Context Analysis

Currently, the similarity between nodes is modelled using the similarity score between the inbound set and outbound set between a pair of nodes. However, it can be further extended into another two groups:

- How similar is the target sink node, and the other sink node originated from the same source node?
- How similar is the target source node, and the other source node with edges pointing to the same target sink node?

Therefore, another two sets of similarity scores reflecting the above two questions are added to the feature set.

5.2. Improvement

Based on the analysis above, the following adjustment are made for the programme:

- Feature Set: Similarity score is calculated for the target sink/source node and the neighbour sink/source nodes. Threshold is set as 100,000
- Similarity Scores: Natural logarithm is applied to similarity scores, including the node degree, its sum/difference and common neighbours. Common logarithm is applied to Priority Link, with consideration to its high value.

6. Conclusions

As indicated in the above report, the traditional approximate string matching method is not sufficient to normalise user generated content such as Twitter data. However, after adjusting the method according to the characteristics of the dataset, it is possible to improve the performance.

References

1. Heikki Hyyro https://pypi.python.org/pypi/editdistance
2. Pablo Gamallo and Jose Ramom Pichel, A Method to Lexical Normalisation of Tweets.
3. Bo Han and Timothy Baldwin (2011) Lexical normalisation of short text messages: Make sense a #twitter. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, USA. pp. 368–378.
4. Jeremy Nicolson, Justin Zobel, and Karin Verspoor (2017) Approximate String Matching.
5. Justin Zobel, Philip Dart, Phonetic String Matching: Lessons from Information Retrieval.
6. Muhammad Marwan, Muhammad Fuad, Towards Normalisation the Edit Distance Using a Genetic Algorithms-Based Scheme
7. Bergstra, J. and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(Feb), pp.281-305.