École Polytechnique

INF554 PROJECT REPORT

Machine Learning 1

Links Prediction in Citation Network

Authors:
Fabrizio Indirli
Leon Kloten
Martin Wohlfender
Seongbin Lim

Team name:
4our

January 2, 2019

Links Prediction in Citation Network

TEAM 4our
Fabrizio Indirli (*fabrizio.indirli@polytechnique.edu*), Leon Kloten (*leon.kloten@polytechnique.edu*),
Martin Wohlfender (*martin-samuel.wohlfender@polytechnique.edu*),
Seongbin Lim (*seongbin.lim@polytechnique.edu*)

***Abstract*—The project for the INF554 Machine Learning course consisted in predicting missing links in a citation network. To do so, 17 features have been extracted from the given data and various supervised learning models have been developed, obtaining an F1 score of almost 0.97.**
Among all the features, 6 of them largely affect the predictions performance: *number of paths*, *TFIDF cosine similarities of abstracts*, *source hub score*, *target authority score* and the *Resource Allocation index*.
Between the various classifiers that were tried, *XGBoost* and *Neural Network* produced the best results.

I. Introduction

SCIENTIFIC papers often cite other articles in their bibliography. Given a set of papers that cite each other, it is possible to build a graph whose nodes represent articles and whose edges are the citations: the edge *(i,j)* belongs to the graph if article *i* cites article *j*; a graph of this type is called a **citation network**. When studying a set of papers concerning the same fields, these networks can be very dense and can carry lots of information for link prediction, clustering, classification, and other tasks.
In the INF554 Project, some edges have been removed from a citation network constructed upon a dataset with 27770 articles: the aim of the project is to build a model that can predict whether an edge was part of the original network or not.
The problem of **links prediction in a citation network** has been widely studied in literature because of its importance: as stated by Shibata, Kajikawa, Sakata, these predictions *"may help scholars to know which paper to cite and managers to identify future core papers"* [1], but other possible applications may be clustering of papers to organize them or to extract statistics on them, improvements on search engines for articles, and much more.

II. Data Analysis

By performing some analysis on the dataset, some useful points have been discovered, which will be covered in the following paragraphs.

A. Some articles cite papers in the future

As shown in Fig. 1, in almost 20% of the citations in the training set the *cited paper* was published after the *citing article*, which should be virtually impossible. In this case, 3 hypotheses have been formulated:

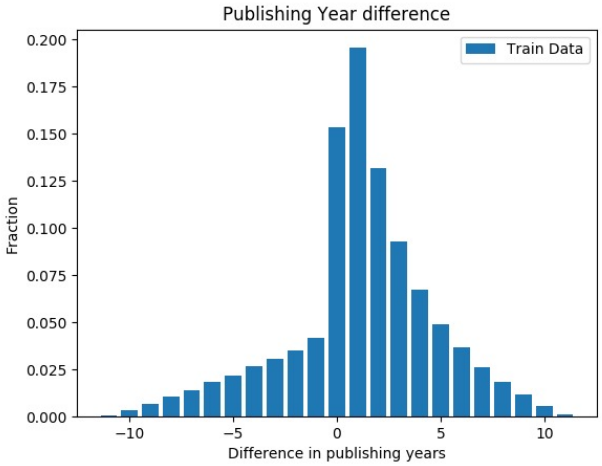

Fig. 1. Distribution of Temporal Difference feature.

- This citation network should be intended as an *undirected graph*; however, the presence of several **duplicated edges** in the undirected graph and the fact that in the literature a citation network is usually a directed graph lead to the rejection of this hypothesis.
- The direction of the edges should be inferred from the sign of the *Publishing Years difference*. However, implementing this solution had a negative impact on the prediction score; in addition, there isn't a clear method to decide the direction of edges whose *Publishing Years difference* is 0; hence, this hypothesis was rejected, too.
- The citation in advance are due to the fact that the authors may know each other and may cite other articles before they are officially published on a journal.

III. Features

For each edge *(i,j)* of the citation network, various features are extracted from *topological properties* of the involved nodes/edge or from *intrinsic properties* of the involved articles. The features are the following:

A. Titles overlap

Number of identical words of the titles of the two documents

B. Number of common authors

Number of common authors of the two documents. Before calculating this feature, the authors are preprocessed to remove

parentheses (which often contain unnecessary information) or special characters (which are often typos).

C. Temporal difference

The difference between the second article's year and the first article's year. This feature is extremely important for a good prediction: most of the linked articles in the training set have a *temporal difference* between 0 and 4 years (Fig. 1).

D. Abstracts TF-IDF similarity

Cosine similarity of term frequency – inverse document frequency (tfidf) vectors extracted from source and target abstracts.
For each paper *p*, a TF-IDF vector w_p is calculated:

$$w_p[i] = tf_{i,p} \cdot \log \frac{numPapers}{df_i}$$

where df_i is the number of papers' abstracts containing the i-th word.
Then, for each edge *(i,j)*, the **cosine similarity** is calculated by multiplying the two vectors:

$$cosineSim(i,j) = w_i \cdot w_j$$

We use the *TF-IDF* similarities instead of a simple common terms count to reduce the weight of common terms that may not carry any meaningful information.
We have also tried to use **Cosine similarities of LSA of TF-IDF abstract**, since LSA (Latent Semantic Analysis) is a common way to deal with TF-IDF feature. It is also known as *truncated singular value decomposition*, thus it is a dimensionality reduction technique.
TF-IDF has a dimension as large as the number of words in all documents, thus there may be problems of synonymy and polysemy, that LSA may solve [5]. This model was fitted with the number of components set to 100 and gave around 0.187 as a sum of explained variance. In the end, it was not used as one of the final features because the total explained variance was too small to be more useful than the standard cosine similarity of TF-IDF.

E. Same journal

A boolean feature that is equal to 1 if the two papers have been published on the same journal. We thought that authors who publish on a journal, also read the same journal and hence have an higher probability of citing articles from that journal; however, this feature seems to have a very marginal impact on prediction performance.

F. Source authors to target journal citations

Sum of the number of citations that the authors of the source paper have made to articles published on the target paper's journal.

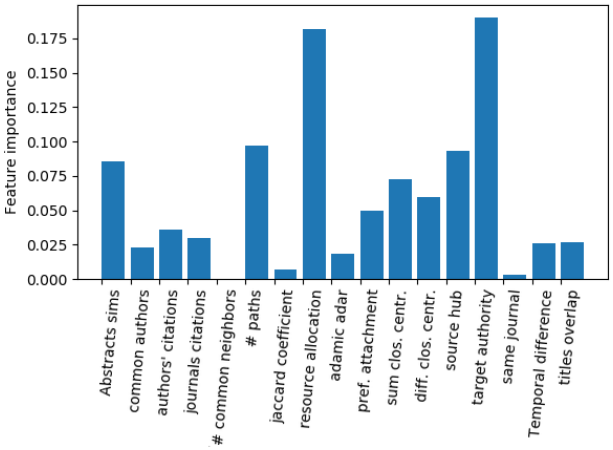

Fig. 2. Relative features importance in link prediction.

G. Source Hub & Target Authority

These values are calculated using the HITS algorithm (*Hyperlink-Induced Topic Search*), which is a link analysis algorithm that was originally developed to rank webpages. In our project, we have used it to calculate, for each edge *(i,j)*:

- **Hub Score of source *i***: estimates the value of links from *i* to other nodes
- **Authority Score of target *j***: estimates the value of paper *j* by valuing the *Hub Scores* of other papers citing *j*.

H. Number of common neighbors

Number of nodes that are directly linked to both *i* and *j*.

I. Link-based Jaccard coefficient

A metric that normalizes the number of common neighbors over the number of total neighbors: this gives and index of similarity over common neighbors that doesn't penalize nodes with less neighbors. For an edge *(i,j)*, the Jaccard coefficient is:

$$Jac(i,j) = \frac{|Neighbors_i \bigcap Neigbors_j|}{|Neighbors_i \bigcup Neigbors_j|}$$

J. Sum and Difference of Closeness centralities

The **closeness centrality** (or closeness) of a node is a measure of centrality in a network, calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes. For a node *x*, it is computed as follows:

$$C(i) = \frac{1}{\sum_{\forall j} distance(i,j)}$$

where *distance(i,j)* is the legth of the shortest path between *i* and *j*.
For each edge *(i,j)*, we calculate two features:

- $$SumCentrality(i,j) = C(i) + C(j)$$

which takes into account how central the two nodes are in the graph, assuming that the probability of an edge existing between two nodes of high centrality is higher than the probability of an edge existing between two nodes of less high centrality.

- $$DifCentrality(i,j) = C(i) - C(j)$$

  which measures how different the two nodes are in terms of centrality

K. Source authors to target authors total citations

Each paper has at least one author. Fore each edge *(i,j)*, this feature calculates the total number of citations that the authors of article *i* have made to the authors of article *j*, considering the entire dataset.
To calculate this feature, an *m* x *m* matrix **M** is calculated, where *m* is the total number of different authors in the dataset and *M(i,j)* is the number of citations that the author *i* has made to the author *j* in the entire dataset.
Then, for each edge *(i,j)*:

$$Cits(i,j) = \sum_{a \in Auths_i} (\sum_{b \in Auths_j} M(a,b))$$

L. Resource allocation index

This similarity feature is inspired by the resource allocation process taking place in networks [2]. Each node *i* is considered as a producer of a resource; for a pair of nodes *i* and *j* non directly connected, the transmitter *i* can send part of its resource to *j* by evenly transmitting it through their common neighbors.
Using this model, for each possible edge *(i,j)* we calculate the similarity based on the resource allocation index:

$$s(i,j) = \sum_{w \in (\Gamma(i) \bigcap \Gamma(j))} \frac{1}{\Gamma(w)}$$

M. Preferential attachment

This feature is based on the assumption that a paper that is often cited in the training set, is probably going to be cited even more in future papers, while articles that have few citations are probably going to be forgotten. The preferential attachment score for an edge *(i,j)* is computed as follows:

$$Pas(i,j) = |\Gamma(u)||\Gamma(v)|$$

where $\Gamma(w)$ is the set of *w*'s neighbors.

N. Adamic/Adar Index

The Adamic/Adar index is a measure introduced to predict links in a social network, according to the amount of shared links between two nodes [3].
For an edge *(i,j)*, it is computed as follows:

$$aa(i,j) = \sum_{w \in (\Gamma(i) \bigcap \Gamma(j))} \frac{1}{\log(\Gamma(w))}$$

where $\Gamma(w)$ is the set of *w*'s neighbors.

O. Number of paths

For an edge *(i,j)*, this feature is the number of simple (acyclic) paths from *i* to *j*.

IV. Models

In order to try to achieve the best predictive performance possible, various model have been tried.

A. Linear regression

Linear regression is the simplest classifier chosen to compare with the others. It assumes the relationship between dependent and independent variables to be linear.

B. Random forest

Random forest is an ensemble algorithm making use of bagging technique and decision tree as a base classifier. It is very robust against overfitting and it is easy to tune comparing to other classifiers because it has only two parameters [4]. Since training random forest model was fast and reliable, it was used as a practical estimation tool to make a quick judgement if new features were effective enough to improve the result of final prediction.

C. Boosted Trees (XGBoost)

The XGB classifier is based on gradient boosted decision trees. As it is a very strong classifier that produced good results in many applications, this method was used as main model.
In XGBoost, boosting is applied on a trees ensemble with an incremental policy: in the training process, instead of assigning different weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss when adding the latest prediction: hence, the model is updated using *gradient descent*.

D. Neural Network

Neural networks are best known for recognition tasks. They have multiple layers, which allow networks to learn multiple levels of abstraction [6]; they are one of the state-of-art algorithm and they have many potentials because there are many hyperparameters to tune when building a model.

E. Other classifiers

A few additional classifiers were used with standard parameters as baselines. these include:

- One Versus Rest classifier (with Linear Regression)
- KNeighbors classifier
- Decision Tree classifier
- Support Vector Machine

V. Testing and Results

A. Linear model

The predictions with the linear model obtained an F-1 score of 0.911154 on test dataset. Meanwhile, the coefficient of determination (R^2) turned out to be around 0.57, indicating that the linear regression model can explain 57% of the variance. We concluded that linear regression model is not suitable for our task based on this value.

B. Random forest

The parameters were tuned by varying the number of estimators (decision trees) and maximum depth of each tree. To make it more reliable, 5-fold cross-validation was employed to validate F-1 score. Even though the deeper the maximum depth got the higher the F-1 score it reached, it seemed to be saturated above 16 depths. Therefore, max_depth parameter sets to be 8 to avoid overfitting. The number of estimators was selected rather small not because it may overfit the model, but because it increased training time a lot. As a result, it gave 0.96544 F-1 score with the number of estimators being 40 and max_depth being 8.

C. Boosted Trees (XGBoost)

Using the XGB classifier a submission that received 0.96973 F-1 score, our best result, was produced. 20% of the training data was used as *validation set* and an **F-1 score** evaluation metric was implemented and used. To prevent overfitting, the *early stopping* parameter was set to 20 rounds. For all but one of the other parameters (*number of jobs = -1*) the default values were left untouched.

Some further effort into parameter tuning did not give a better result. By changing the number of estimators, the learning rate and the maximal depth no better score could be achieved on a submission. This was unexpected because an F-1 score of more than 0.975 was achieved on average on a 5-fold cross validation with the following choice of parameters: number of estimators = 100, learning rate = 0.3 and maximal depth = 5.

D. Neural Network

There were trials to tune neural networks, mainly varying the number of hidden layers, the number of neurons at each layer, and the learning rate of Adam optimizer (Fig. 3). Furthermore, in order to prevent overfitting, many techniques were used, from simple validation splits with early stopping and checkpoint to dropouts. We managed to achieve an F1 score 0.96839 by tuning without the overfitting-avoidance techniques described above, which was higher than the result 0.96600 from tuning with overfitting-avoidance. We concluded that those techniques led the model to be slightly underfitted.

E. Other classifiers

The following F-1 scores have been achieved on a 5-fold cross validation using default parameters:

- One Versus Rest classifier (with Linear Regression): 0.95144
- KNeighbors classifier: 0.96073
- Decision Tree classifier: 0.96088
- Support Vector Machines: 0.87875

VI. Conclusions

According to our tests, the **Boosted trees (XGBoost)** model achieved the best results with an F1 score of almost 0.97. This was not unexpected due to the many qualities of the XGBoost classifier, such as the ability to avoid overfitting by checking the model on a validation set or the fact that it takes the bias-variance tradeoff into consideration during fitting. We also concluded that the most important features for this kind of predictions are *number of paths*, *TFIDF cosine similarities of abstracts*, *source hub score*, *target authority score* and the *Resource Allocation index*.

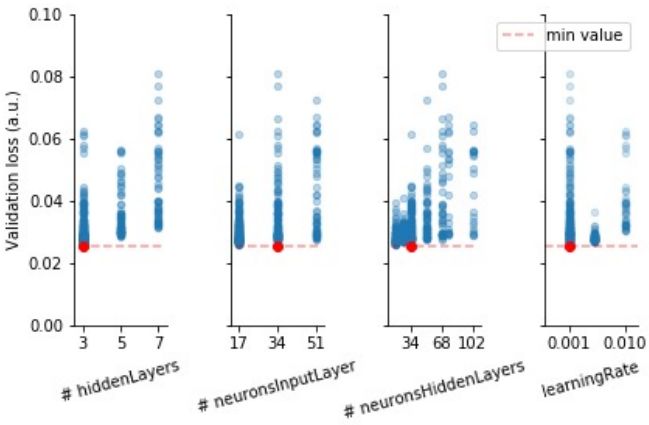

Fig. 3. Validation loss value of models during the tuning process. Each dot represents one model. Note that the tuning was multivariate process, thus every variation was occured at the same time. .

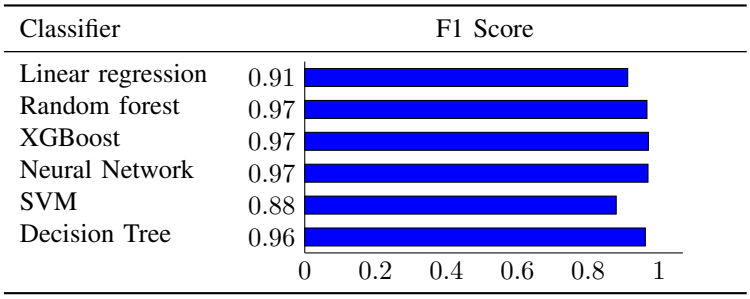

Fig. 4. Comparison of various classifiers performances.

References

[1] N. Shibata, Y. Kajikawa, I. Sakata. Link Prediction in Citation Networks. *Journal of the American Society for Information Sciende and Technlogy* vol. 63, iss.1, pp. 78–85

[2] T. Zhou, L. Lu, Z. Yi-Cheng. Predicting Missing Links via Local Information. *Eur. Phys. J. B*, vol. 71, pp. , 623–630, 2009

[3] L.A. Adamic, E. Adar. Friends and neighbors on the Web. *Social Networks*, vol. 25, iss, 3, pp. 211-230

[4] A. Liaw, M. Wiener. Classification and Regression by randomForest. *R news*, Vol. 2/3, pp 18-22, December 2002, ISSN 1609-3631

[5] Christopher D. Manning, P. Raghavan, H. Schtze (2008). *Introduction to Information Retrieval*, Cambridge University Press, chapter 18: Matrix decompositions & latent semantic indexing.

[6] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. *Nature*, volume 521, pages 436444 (28 May 2015).