A Comparative study of different Machine Learning techniques on Kaggle Twitter Social Influencers Data Set

Arora, Pragya*, Ghai, Piyush†, Gupta, Harsh‡ and Ramkrishnan, Navnith§
Department of Computer Science & Engineering, The Ohio State University
Columbus, OH 43202
Email: *arora.170@osu.edu, †ghai.8@osu.edu, ‡gupta.749@osu.edu, §ramkrishnan.1@osu.edu

***Abstract*—Social media like twitter has given advent to growing importance to understand the behavior of these networks. More importantly these graph networks shows different patterns which are interesting to analyze. Some nodes becomes almost a sink where as some node acts like a source of outgoing links. Understanding these patterns will help in various applications and specially in experiments like Network A/B Testing where we need to introduce some new features to a treatment group restricting the control group. In out project we intended to experiment and analyze on one such pattern where we try to identify which node is more influential when compared to another node. To understand such a behavior we used a Twitter network data set which had 11 features for each node and we applied machine learning techniques to understanding the underlying features to classify the task at hand.**

I. Introduction

The growing importance of Social Networks has given an impetus to understand more about its implications because networks like these can influence millions of people. Contrary to paper or television media, social media is far more reaching to the audiences across the world and information exchange has never seen such a revolution. The need of the time is to have an The ability to understand the nuances of these networks because with more connectivity it becomes crucial to understand the behavior of different nodes in the network.

Some nodes/users are more important than others as there behavior affects a larger sets of nodes in the graph. When a user of such a graph is using the network she/he may find some activity shared by some of the other nodes more striking than others. A lot of psychological and cognitive science study shows that ones life is greatly impacted by her/his immediately surrounding and their influence. As social network has become a part of everyone's life, it has become important which are the nodes which has more impact on others i.e which are influential. One application of such a study is to predict what is the general mood of the crowd during election by analyzing the social network handle of all the candidates who stood for it.

In this project, we address a targeted challenge of this problem presented in a Kaggle competition from 2013 , in which influence on Twitter is predicted from a set of features derived from user activity on the network, such as follower count, retweets received, etc. Rather than identifying influencers in the larger network, we are asked to simply identify which of a pair of users, A and B, is more influential, based on information about their respective activity in the network. This is a binary-classification problem. To find an optimal prediction method, we applied pre-processing methods including feature transformations and dimensionality reduction via principal component analysis (PCA). We used four different machinelearning algorithmsLogistic Regression, Support Vector Machines (SVM), Neural Network, and Gradient Boostingto model the data. We also tried varying some of the decision boundary options for the SVM by using different kernels and neural network classifiers by varying the activation functions on the hidden layer. Our approach includes a few methods that we did not see in other attempts at this problem, specifically a binary feature transformation, the application of principal component analysis, and gradient boosting. We found the most effective pre-processing method to be a logarithmic transformation of the data, while the most successful model was built with gradient boosting, which achieved an 0.869 AUC using a logarithm transform. Outside of gradient boosting, the model configurations which produced the best results used linear-like decision boundaries.

II. DATASET

This section expands on the data set which we used to try the different models on.

A. Original Dataset

The data set which we used came from a Kaggle contest. The data set had a

- train.csv
- test.csv
- sample_predicitions.csv

The train.csv file contains data points where each data point is a pair wise features for two users along with a binary value which reflects which user is more influential. If the value is 0 the first user is more influential and vice-verse. Each user has 11 unique features which are extracted from their twitter profile and recorded in numeric form. The 11 features extracted includes:

- Follwers
- Follwing

- Listed
- Mentions received
- Re-tweets received
- Mentions sent
- Re-tweets sent
- Posts
- Network Feature 1
- Network Feature 2
- Network Feature 2

Similarly we have features for two users in each data item in the testing file but the testing file has no class label. There are 3000 training samples and 2500 testing samples in our data set. Given a test sample, our job is to predict which individual in this test sample is more influential.

B. Data Preprocessing

Since the data is a numerical data, we use some pre-processing techniques on it. The dataset consists of 22 attributes which is essentially 11 attributes for each user in contention. Since we have to compare who is more influential in the network, we subtract related attributes for each user, to reduce the dimensionality to 11. The following is the transformation applied :

$$a_j = x_j - x_{j+11}, j = 1, 2, 3...., 11.$$

Since the attributes are of a different scale, we also apply a logarithmic transformation on the data. The following is the transformation applied :

```
def transform_features(x):
    return np.log(1 + x)
```

Listing 1: Log Transformation

All of these transformations are applied to all the models used.

III. SYSTEM MODELS

A. Baseline

For baseline, we tried a dumb baseline model, where we classified an influencer based on the number of followers, i.e. if X has more followers than Y, then X is more influential. Using this dumb baseline, we get an accuracy of about **70.2%**. This result shows that the number of followers is a strong indicator of the influencers. However, 70.2% is not a very satisfying result and hence we use it only as a benchmark. We know further experiment with more models and a bit more pre-processing.

B. Logistic Regression

First, we preprocess the original dataset using the previous method. After preprocessing the original dataset, different attributes have different ranges which vary a lot. Thus the first thing to do is to handle the data to make it more uniform and easy to deal with. So in order to achieve this, we have a normalization on the dataset. For each feature, we do the following normalization z (i) j = z (i) q j Pm i=1 z (i)2 j , z = 1, 2, ..., n (4) where m is the number of training examples and n = 11 is the number of features. Because our job is to predict who is more influential given two persons, the number of followers, as we know, plays a significant rule in the prediction. A person with more followers than the other is more likely to be judged influential by users. So we will multiply the normalization of the first feature, i.e. number of followers, by a positive constant factor which is greater than 1 to make it more influential than other features on the prediction. For this logistic regression problem, we have two choices gradient ascent and Newtons method to achieve the parameter . Since the number of features n = 11 is not very large, so it will not take much time to compute the inverse of a nn matrix. Thus Newtons method should converge faster in this problem and we choose Newtons method as our implementation. The update rule of the Newtons method is: := H 1l() (5) where H R nn is the Hessian matrix and Hjk = Pm i=1 h(z (i))(1 h(z (i)))z (i) j z (i) k , l() = Z(y h(Z)). Furthermore, we will add cross validation and feature selection to this model, which will be discussed in details in section IV. We write our own code to implement the Newtons method.

C. SVM

We applied the technique of SVM's with a linear kernel and the results were on par with those of Logistic Regression with an AUC of x. Since the original feature set consisted of 11 features, intuitively mapping them to a higher dimensional space wouldn't improve performance. Thus we preferred the linear kernel over the non-linear kernels since it is a simpler model and hence less prone to over-fitting.

Moreover, we add cross validation and feature selection to this model, which refers to section IV.

D. Gaussian Naïve Bayes

Logistic regression and SVM are discriminative learning algorithms. Gaussian Naïve Bayes, is a generative model. The distribution of the original data is not Gaussian, so we tried Z-Score normalization in order to make it as a Gaussian distribution. In Z-Score, we do the following for every value :

$$z = \frac{x - \mu}{\sigma}$$

This converts the data into 0 mean and deviation 1. We then used Gaussian Naïve Bayes from *sklearn* package.

E. Neural Network

We wanted to explore and see how the date set behave in a non-linear environment. The idea was to compare the traditional techniques alongside some state of the art machine learning models which have gained a lot of importance today. Hence we implemented a basic Neural Network model in Python to see how it behaves in this classification task. Our model was customizable which means we could add or remove any number of hidden layers with any number of units in each of them. To our surprise the model worked well but didn't outperform the traditional model. The intuition behind this is that the data set and the features we have are linearly separable.

IV. Tuning the Models

A. Feature Selection

Since our dataset has 11 features, it is quite possible that there are some features which are not quite useful. We used forward selection algorithm to zero in on the best features in order to improve the test accuracy. We used Logistic Regression model in order to select the best accuracies for feature selection. The best features are as follows :

- Follower Count
- Listed_Count
- Retweets Received
- Network Feature 1
- Network Feature 2

B. Cross Validation

We use held out cross validation, where we partition 80% of the data as the training set and 20% of the data as the test set.

V. Results & Discussions

A. Evaluation Criteria

B. Logistic Regression & SVM

In this section, we apply cross validation and feature selection to the linear models, i.e. Logistic Regression and SVM. The following figures compare the performance among different models. In Fig. 2, we use different cross validation options on linear models, such as hold-out and k-fold, Fig. 2. Test accuracy vs. cross validation options for linear models and compare their test accuracies. It is shown that cross validation can improve the test accuracy of SVM, while having no strong effect on Logistic Regression. Whats more, the performance of SVM is better than Logistic Regression using cross validation. Fig. 3. Test accuracy vs. discretization parameter for linear models Fig. 3 shows the test accuracies of SVM and Logistic Regression with different numbers of features selected. As is shown above, the performance of SVM is better, compared with logistic regression. The best performance of SVM is achieved when 4 features are selected, which shows that some features are weak indicators. After feature selection, we sort the features from the most relevant to the least relevant as follows: SVM 9 6 7 10 8 5 3 2 1 4 11 LR 3 6 8 2 9 5 4 7 1 11 10 where the corresponding feature name refers to section II-A. From this table, we can see that although the sort results shown above are quite different for these two methods, their test accuracies are very close. Using linear models, the best accuracy we can achieve is 76.002 and Fig. 3).

C. Naive Bayes

In this model, we compare the performances of different discretization parameters. Here we assume that all features have the same number of classes (the coordinate ascent based algorithm takes too much time). We also mentioned in section III-D that logarithm function is used to preprocess the data. 2 3 4 5 6 7 8 9 10 0.62 0.64 0.66 0.68 0.7 0.72 0.74 0.76 0.78 0.8 # Classes of each attribute Accuracy With Logarithm Without Logarithm Fig. 4. Test accuracies vs. number of classes of each feature In Fig. 4, we also compare the performance with logarithm preprocessing and the one without logarithm processing. We can see that logarithm processing helps increase the test accuracy and when the number of classes of each feature increases, their performances approaches. In addition, it turns out that different discretization parameters dont have much impact on test accuracy. In this case, the best test accuracy is 76.48

D. Neural Network

In this model, we use 50 neurons in the hidden layer. We also apply hold-out cross validation with 30Operating Characteristic) curves are shown in Fig. 5. After training the two-layer network, we get the test accuracy 79.04area under the test ROC curve. 0 0.2 0.4 0.6 0.8 1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 False Positive Rate True Positive Rate Training ROC 0 0.2 0.4 0.6 0.8 1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 False Positive Rate True Positive Rate Validation ROC 0 0.2 0.4 0.6 0.8 1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 False Positive Rate True Positive Rate Test ROC 0 0.2 0.4 0.6 0.8 1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 False Positive Rate True Positive Rate All ROC Fig. 5. Receiver operating characteristic

VI. CONCLUSION & FUTURE WORKS

From the above results, we can conclude that the best accuracy we can achieve is about 76linear models, which is not much better than our benchmark 70.2examples might not be linearly separable. The test accuracy of Naive Bayes is close to linear models. However, if we can apply the coordinate ascent based algorithm, we may achieve much better performance, which is a good choice for future works. Furthermore, the accuracy of nonlinear models such as Neural Network is better than linear models, although its not as good as we expected. Moreover, there might be data corruptions since sometimes human judgement can be highly biased. In order to achieve better performance, we can either try more nonlinear models or use decision trees to improve it.

References

[1] The ISEAR Dataset is available from : http://emotion-research.net/toolbox/toolboxdatabase.2006-10-13.2581092615

[2] SentiWordNet is a lexical resource for opinion mining. More about it can be found at : http://sentiwordnet.isti.cnr.it/

[3] S. Mostafa Al Masum, H. Prendinger and M. Ishizuka, "Emotion Sensitive News Agent: An Approach Towards User Centric Emotion Sensing from the News," Web Intelligence, IEEE/WIC/ACM International Conference on, Fremont, CA, 2007, pp. 614-620. doi: 10.1109/WI.2007.124

[4] Word2Vec tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research. https://code.google.com/archive/p/word2vec/

[5] GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. https://nlp.stanford.edu/projects/glove/

[6] The following is a good link about understanding LSTMs : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] A Keras implementation of using pre-trained word embeddings. https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

[8] Yoon Kim, Convulational Neural Networks for Sentence Classification, http://arxiv.org/abs/1408.5882

[9] An implementation for CNNs for Sentence Classification can be found at : http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/