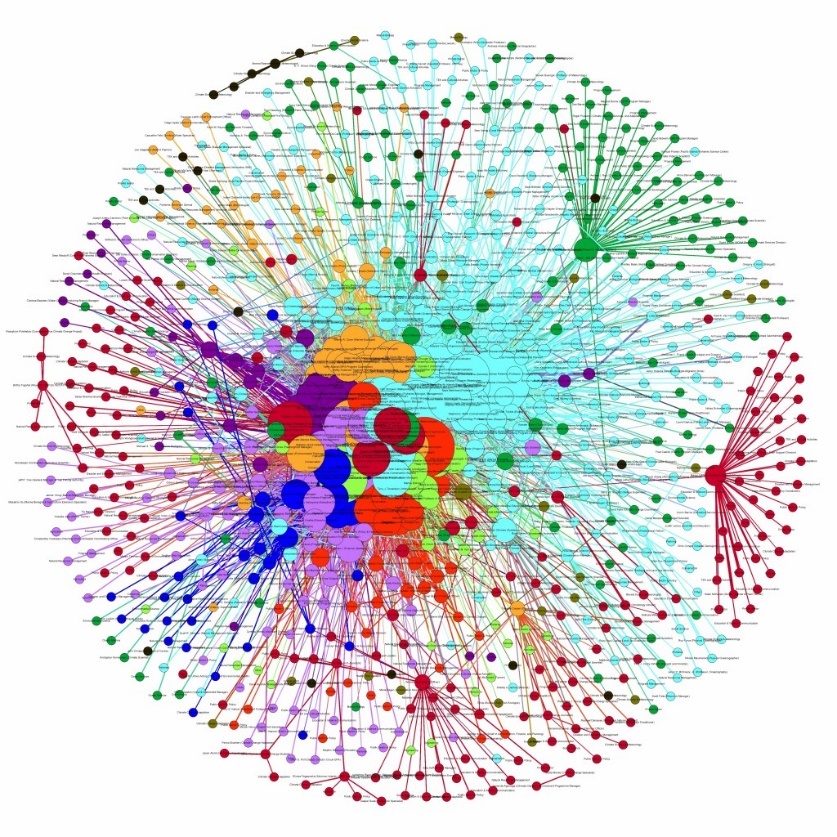
# Data Science Project Report

## This is a brief report of what Nicholas and I have done so far for the Data Science Seminar project. The project consists of using force-directed graph drawing algorithms to improve the predictive power of machine learning classification tasks when using network data. Unfortunately, initial results show that other benchmarks perform better than our proposed method.

## Summary

Inspired by the work we read in class by Fang, Hu, Li, & Tsai (2013), our proposal seeks to improve the performance of classification algorithms in network contexts using the positions of each instance within the network. Most relevant social network theories emphasize that nodes that are closer to each other (i.e., that are fewer edges apart) have a higher probability of being similar for various reasons, such as social influence or homophily. Hence, network analysis in classification problems usually involves quantifying in some way the distance between unlabeled nodes and labeled nodes, with the hope that this distance will be predictive of the unknown labels. However, the network analyses we’ve seen in class usually account only for the immediate neighbors of each node, since making a full-fledged analysis implies an exponential increase in computational time and would be unfeasible in most scenarios. But what if we could use information about the neighbors’ neighbors? Or about the neighbors of the neighbors’ neighbors?

According to social network theory, if we draw a labeled network in a two-dimensional plane, we would expect nodes with the same label to clutter closer with each other than with nodes with other labels. If each label had a color, we would expect to see patches where certain colors are concentrated instead of seeing an uninterpretable multicolored mess. That is, we would expect to see something like this:



Obtained from: <http://www.pacificrisa.org/projects/social-network-analysis/full-network/>

In such a drawing, we would expect the “geographical” position of a node to be informative of its label. This is where our proposal comes in: using the geographical positions generated by force-directed graph drawing algorithms (FDGs) as features for machine learning algorithms. FDGs are a class of algorithms that is used to draw graphs (a network being a type of graph) in an interpretable and usually aesthetically pleasing way. Their purpose is to position the nodes of a graph in a two-dimensional or three-dimensional space so that all the edges are of similar length and there are as few crossing edges as possible. While there are many different types of FDGs, all of them work by assigning forces among the set of edges and the set of nodes based on their relative positions. Then, these forces are used to reproduce a physical simulation (e.g., the motion of the edges and nodes) and subsequently determine the final position of the nodes. While graph drawing can become a difficult problem, FDGs rely only on information contained within the structure of the graph itself and require no domain-specific knowledge. In a way, you could think of FDGs as a “black box” that receives a graph as an input and returns a pair of coordinates for each node as an output. Our proposal is that these coordinates could be used as additional features to improve the predictive power of classification algorithms in network contexts.

## Data

We’re working with Facebook data retrieved from the Stanford Large Network Dataset Collection, a public database library from Stanford University that has been collected (scraped) to research large social and information networks. The dataset consists of the profile information and the friend lists of Facebook users and was collected from survey participants using a Facebook app called ‘Social Circles’. The data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, the interpretation for each feature has been obscured. For instance, where the original dataset may have contained a feature "gender = male ", the new data would simply contain "feature 10 = 1". Thus, it is possible to use the anonymized data to determine whether two users share the same feature value, but not to determine what each individual feature represents.

The data consists of 10 ego networks. In turn, each ego networks consists of a focal node (“ego”), the nodes to whom ego is directly connected to, and all the edges among the nodes included in the network. Summing up all networks, there are 4,039 nodes and 88,234 edges. However, our analyses so far have been done exclusively in one of the ego networks, which has 1,045 nodes and 53,498 edges. In addition, each node has 576 obscured binary features.

## Work So Far

We wrote a small program in R that benchmarks the AUC of our proposed method. The inputs of the program are the network data (i.e., the lists of nodes and edges) and a binary node-level feature (i.e., the target variable). The program outputs the AUC of three different methods, using the average result of a 10-fold cross validation. Folds were created using stratified samples.[[1]](#footnote-1)

* **FDG Method**: This method uses the force-directed layout algorithm by Fruchterman and Reingold to calculate the geographical positions of each network node in a two-dimensional space. We chose this specific algorithm because it is one of the most popular to draw graphs, but there are many other FDG options. Then, we train an SVM model using the two geographical positions as covariates and the provided binary feature as the target variable. The scores generated by this model are what we use to calculate the AUC of this method.
* **Counting Neighbors**: For each node, we calculate a score by summing up its positive neighbors (i.e., all nodes that are one edge away and have a value of 1 in the binary feature). We use these scores to calculate the AUC of this method.
* **Combined**: This method is the same as the “FDG Method”, but it also includes the “Counting Neighbors” score as part of the SVM model features.
* **Floyd-Warshall**: We use the Floyd-Warshall algorithm to find the shortest paths from all nodes to all nodes. For each node, we calculate a score by summing up the shortest distances to all positive nodes (i.e., the nodes that have a value of 1 in the binary feature). We then multiplied these scores by -1. We used these scores to calculate the AUC of this method.

We use these methods to predict each of the binary features in the network. Therefore, we run 576 tests in total (i.e., 576 AUC results for each method).

## Results

The graph below presents a summary of the performance of each method for the 576 target variables.

Graph 1: Performance of the 3 methods for all target variables

As shown in the results table below, “Counting Neighbors” (red) outperforms the other three methods most of the time. A surprising fact is that combining both the proposed method and the “Counting Neighbors” method harms the performance of both.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | FDG | Counting Neighbors | Combined | Floyd-Warshall |
| Average AUC | 0.70 | 0.80 | 0.50 | 0.53 |
| Average Rank (1 being the best, 4 the worst) | 2.27 | 1.44 | 3.19 | 3.10 |
| Times ranked as 1st | 131 | 386 | 34 | 26 |
| Times ranked as 2nd | 252 | 141 | 72 | 110 |
| Times ranked as 3rd | 96 | 37 | 222 | 218 |
| Times ranked as 4th | 95 | 12 | 247 | 219 |

Graph 1: Performance of the 3 methods for all target variables

## Next Steps

At this point, results seem kind of discouraging, so we are unsure of whether we should continue to work on this project or not. To our knowledge, nobody else is working on this, but this may be a sign of why this is the case. We were thinking that we could substantially improve the performance of our proposal if we created our own algorithm using the geographic positions generated by the FDG. An obvious choice would be to calculate the average distance to all positive instances and use this as the score generated by the algorithm. However, we consider that this would probably be incredibly slow, especially because generating the FDG coordinates is already quite computationally intensive. However, we would like to have your input on the project.

## References

Fang, X., Hu, P. J., Li, Z. L., & Tsai, W. (2013). Predicting Adoption Probabilities in Social Networks Predicting Adoption Probabilities in Social Networks.

1. In our data, there are cases where there are too few positive instances to perform a 10-fold cross validation. So, we only use k folds (k being the total number of positive instances in the data) whenever k is less than 10. [↑](#footnote-ref-1)