Application of Machine Learning to Link Prediction - Milestone

Kyle Julian (kjulian3), Wayne Lu (waynelu)

November 22, 2016

1 Background and Introduction

A real-world network is often modeled as a graph G with vertices V and edges E, denoted as $G = (V, E)$. The following terms are often used interchangeably: network/graph, node/vertex, and link/edge. Nodes represent objects in the domain such as people or products, and links represent relationships such as Facebook friendships or the purchase of a product. Links can be either directed or undirected, representing a one-way or mutual relationship, respectively. For example, Twitter follows are one-way while Facebook friendships are mutual.

Edges are notated as a pair of nodes (u, v). In a directed graph, this represents an edge from u to v. In an undirected graph, this represents a bidirectional edge between u and v. Two nodes u and v are called adjacent if there exists either an edge (u, v) or an edge (v, u). The neighborhood of a node u is the set of nodes adjacent to u. A node and edge are called incident if the node is one of the endpoints of the edge. The degree of a node u is equal to the number of edges incident to u, denoted $d(u)$. For a directed graph, the in-degree of a node u is equal to the number of edges pointing into u, denoted $d_{\text{in}}(u)$, and the out-degree of u is equal to the number of edges pointing out of u, denoted $d_{\text{out}}(u)$.

Real-world networks evolve over time as new nodes and links are added. Link prediction algorithms use historical data in order to predict the appearance of a new links in the network or to identify links which may exist but are not represented in the data. The application of link prediction is most commonly seen in recommendation engines, either for connections on social networks or products on shopping sites. Traditional approaches involve the calculation of a heuristic score for a pair of nodes, such as the number of common neighbors or the shortest path length connecting the nodes. In this project, we will apply supervised learning algorithms to the link prediction prediction problem using a large set of topological features. The use of a large set of features allows us to then apply feature selection algorithms to identify the most important features for link prediction. Furthermore, by using only topological features and remaining domain-agnostic, we are able to investigate structural similarities between network.

2 Related Work

Link prediction and the application of machine learning techniques to link prediction both have significant corpuses of work behind them. Liben-Nowell and Kleinberg used a variety of topological heuristic scores as link predictors for arXiv co-authorship networks and compared their relative performances [1]. Al Hasan et al. applied supervised learning to link prediction on the BIOBASE and DBPL co-authorship networks, using both topological and domain-specific features [2]. Leskovec et al. applied supervised learning to edge sign detection in real-world social networks, notably comparing the performance of trained algorithms across different datasets and using the idea of triads as part of their feature set [3].

3 Methodology

3.1 Datasets

The data used in this project are publicly available from Stanford Network Analysis Project (SNAP). Our current datasets consist of the Wikipedia Request for Adminship (RfA) voting network and the five arXiv co-authorship networks used by Liben-Nowell and Kleinberg.

Dataset	**Nodes**	**Edges**
Wikipedia RfA	7,115	103,689
arXiv Astro Physics	18,722	198,110
arXiv Condensed Matter	23,133	93,497
arXiv General Relativity	5,242	14,496
arXiv High Energy Physics	12,008	118,521
arXiv High Energy Physics Theory	9,877	25,998

3.2 Features

Our preliminary feature set consists of 17 topological characteristics: 8 degree features and 9 neighborhood features. Given a pair of nodes u and v, our degree features consist of $d_{\text{in}}(u)$, $d_{\text{out}}(u)$, $d_{\text{in}}(u)/d_{\text{out}}(u)$, $d_{\text{out}}(u)/d_{\text{in}}(u)$, $d_{\text{in}}(v)$, $d_{\text{out}}(v)$, $d_{\text{in}}(v)/d_{\text{out}}(v)$, $d_{\text{out}}(v)/d_{\text{in}}(v)$.

Our neighborhood features draw inspiration from the triads described by Leskovec et al. [3]. For a pair of nodes u and v and a common neighbor w, we identify four different triad types:

Type 0	$u \rightarrow w \rightarrow v$
Type 1	$u \rightarrow w \leftarrow v$
Type 2	$u \leftarrow w \rightarrow v$
Type 3	$u \leftarrow w \leftarrow v$

The participation rate t_n of u and v in a Type n triad is the number of common neighbors to u and v which form that triad. Our neighborhood features then consist of the number of common neighbors $C(u, v)$, t_0, t_1, t_2, t_3, $t_0/C(u, v)$, $t_1/C(u, v)$, $t_2/C(u, v)$, and $t_3/C(u, v)$.

3.3 Example Generation

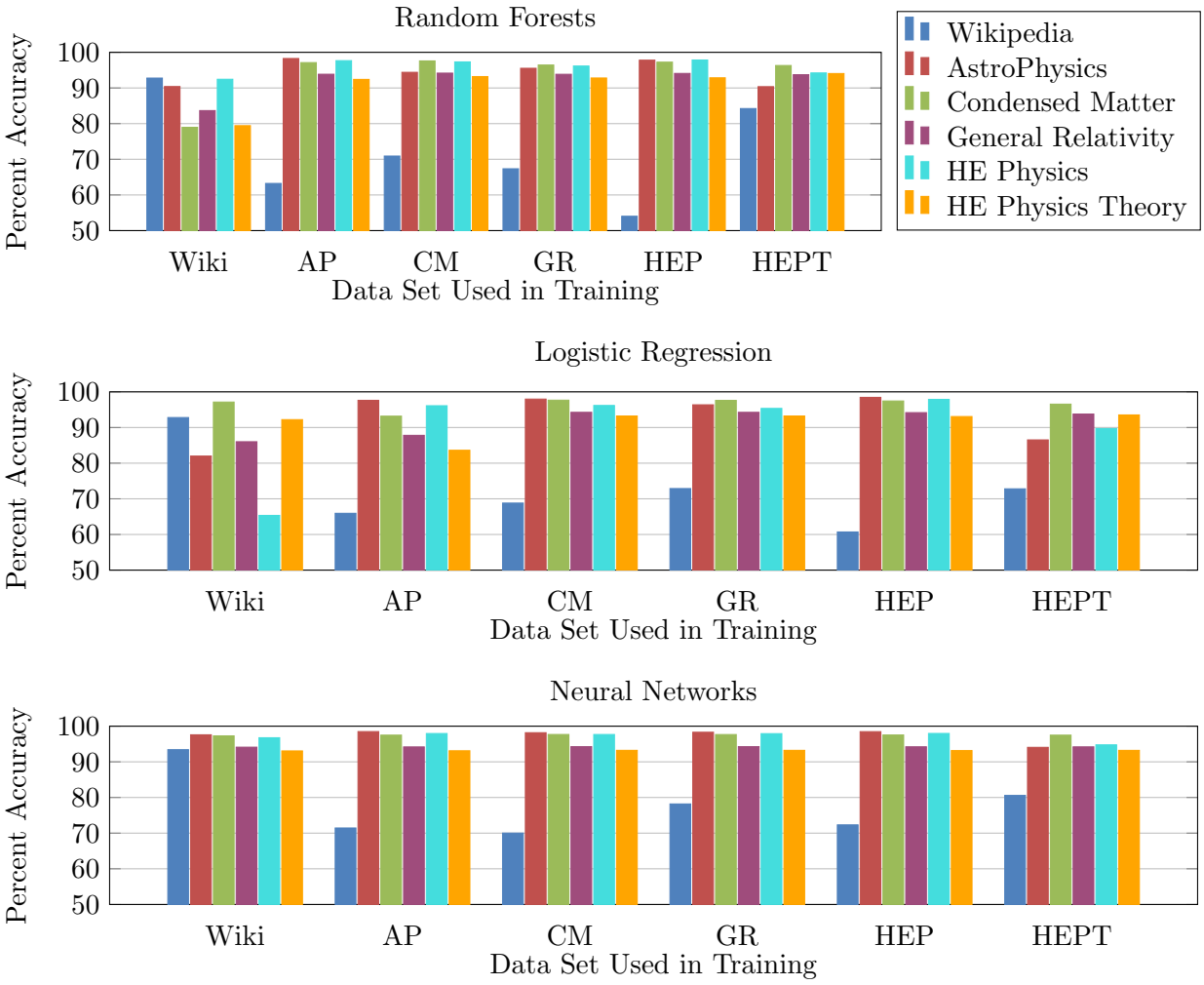
For each dataset, training and testing examples were generated by first partitioning the existing edges into training and testing sets via a 90-10 split. The sets were then doubled in size by supplementing each of them with non-edge node pairs generated by random sampling. Thus, the training and testing examples contain an equal number of edge and non-edge classification examples. Finally, the previously described features were extracted for each example.

3.4 Supervised Learning Algorithms

We used three different classification algorithms: logistic regression, neural networks, and random forests, which are implemented in the Scikit Learn python package. The logistic regression used L2 regularization and limited-memory BFGS optimization, an efficient quasi-Newton method. The neural network classifier is comprised of 5 fully connected hidden layers of 30 nodes each with rectified linear unit activations and cross-entropy loss. The random forests algorithm trains 10 decision trees from random samples of the inputs and then averages the individual decisions to create a single decision for the forest. Because decision trees tend towards high variance, taking the average of randomized trees helps to decrease the variance and improves generalization.

4 Preliminary Results

First, we trained each classifier using the training data from one of the six data sets. Then, we tested the trained classifier on each data set, which tells us the generalization performance of the classifier on different data sets's testing data. The accuracies are shown in the figure below. The results show that there is a strong correlation among the graphs, especially in the co-authorship graphs. Classifiers trained on co-authorship graphs did not perform as well on the Wikipedia voting graph. However, when a neural network was trained on the Wikipedia voting graph, all of the testing sets performed well. Further work will study this correlation as well as the impact of feature selection on the classification performance.



5 Works Cited

1. Liben-Nowell, David, and Jon Kleinberg. "The link-prediction problem for social networks." *Journal of the American society for information science and technology* 58.7 (2007): 1019-1031.

2. Al Hasan, Mohammad, et al. "Link prediction using supervised learning." *SDM06: workshop on link analysis, counter-terrorism and security.* 2006.

3. Leskovec, Jure, Daniel Huttenlocher, and Jon Kleinberg. "Predicting positive and negative links in online social networks." *Proceedings of the 19th international conference on World wide web. ACM*, 2010.