Project Status Report for CIS 419
Selecting the Optimal Shortest Path Algorithm for Road Networks

Abstract

Over 1,000 lives are lost every year as a result of delays in emergency response times. Fast detection of the shortest possible path between a starting point and an emergency destination could contribute significantly towards increasing the speed of emergency responders and thus, their ability to save lives. To achieve this goal, a machine learning classifier can be used to detect important features of a particular road network given source and target, and determine and implement the most efficient shortest path algorithm. We explore the viability of this option in this paper by training three possible classifiers to choose between variations of Djikstra's, Bellman-Ford-Moore, and Gordon Growth shortest path algorithms. Having considered two types of Support Vector Machines and a decision tree classifier, we find that our ultimate choice of XXXXXXX showed promise in most efficiently identifying an appropriate shortest path algorithm to use, such that an appropriate path is output.

1. Project

1.1. Introduction

Current research efforts in the area of shortest path algorithms have focused on specific types of graphs, and suggest cases for when a specific algorithm should be used based on characteristics of the input graph. This means today, users must either know about specific features of the graph to decide which algorithm to use, or they must choose an algorithm at random.

Often, Dijkstra's shortest path algorithm is used since it is the most general, but alternatives may outperform this choice on select input graphs. However, depending on the feature vectors specific to any given instance of a graph, another algorithm may more efficiently produce the shortest path; for the purposes of this project, we considered variations of Djikstra's, Bellman-Ford-Moore, and Graph Growth shortest path algorithms.

We plan to compare the efficiencies of variations of Support Vector Machine and decision tree classifiers, trained on real-world road network data, to determine both the most effective classifier, as well as to triangulate the most efficient shortest path algorithm for an arbitrary road network in working towards the final goal of outputting the optimal shortest path algorithm. The training and test datasets are sourced from the Dryad Digital Repository under the title *Barrington-Leigh-Millard-Ball-PNAS2015-Century-of-sprawl-graphs-in-GML-format*. An overview of our progress thus far and the work remaining to be done is written below.

1.2. Work Completed to Date

A first step in the project was determining the the file format for our graph data. Currently, there is no one universal structure to represent graphs, since often times different applications require different internal graph structures. We ended up choosing Graph Modeling Language (GML) because it offers the following advantages: it is easy to read, understand, and parse. Furthermore, there is a lot of graph data available in this format, and many algorithms and packages have already been defined for this structure. We obtained simple GML graphs from Mark Newman's personal website (**?**) from which to test our parser, and eventually to test our feature extraction.

The next step was to identify datasets on which to train our three classifiers. These datasets needed to be representative of the potential test data that might be passed in to the classifier for predictions after it has been trained. The motivation for the classifiers was to provide emergency services and map services with the fastest algorithm to use to compute the shortest path for a given source and destination, so the classifier would be primarily used with graphs of road data. Therefore, to ensure that our training data and potential test data were drawn from the same overall sample, which in this case is roads, we realized we needed road data for our test data. As stated before, we obtained this data from the Dryad Digital Repository under the title *Barrington-Leigh-Millard-Ball-PNAS2015-Century-of-*

University of Pennsylvania, CIS 419/519 Course Project.

sprawl-graphs-in-GML-format, as specified in the introduction. There were hundreds of graphs, each with thousands of nodes, all in GML format.

Next, we parsed this graph data in order to obtain feature vectors that represent each graph. We used the *NetworkX* package to obtain most of the following features of our graphs and then combined all of the feature vectors into a 2D *numpy* array, where each column represented values of a particular feature and each row was a different graph. We will then pass this numpy array as the X argument in the fit method of our classifier. The features we have computed are:

1. Maximum Degree Centrality
2. Minimum Degree Centrality
3. Maximum Load Centrality
4. Average Clustering
5. Node-to-Edge Ratio
6. Radius

We initially chose to sample three different classifiers to compare their performance, given that our target application was to find the shortest geographical path between two points for emergency responders. We decided to use Support Vector Machines because we expect that these data may not be linearly separable, and SVMs are uniquely capable of targeting such data. Further, we chose to test a decision tree classifier because it can handle the graph features we use as inputs to the base learning algorithm.

We believe it would be a particularly good fit for our purposes because it requires only a relatively general base learning algorithm. Therefore, we know that future test data will conform to the road data we used for testing. It is important to remember that certain classifiers, such as Naive Bayes, would be unsuitable since many of the features we have computed over the graphs may be correlated with each other.

1.3. Work Remaining

First, we need to finish writing the various different Shortest Path algorithms and compute the running times of each algorithm for each graph in our training set, and then assign each graph its true label based on which algorithm runs fastest.

We also still have to finalize our choice of classifier based on our target dataset. Although we have used the ADABoost classifier up to this point, we will sample other classifiers with our base learning algorithm to determine the best possible pairing. Once we have decided on a classifier, we will need to tune its parameters using GridSearch over the training set based on the true labels of the training set.

Having chosen this classifier, we will compare the prediction results of our classifier on our training set with the true labels for the training data. This comparison will enable us to obtain evaluation metrics including accuracy, precision, and recall, against which to score our classifier and determine its viability as a solution for emergency responders.

We will then compile a series of various figures, such as ROC curves, to display the performance of our algorithm on various test instances. Beyond just the accuracy of our classifier, an important metric for the application we are considering is runtime, so we must take this into account when deciding on a specific classifier.