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

Abstract

We implemented a classifier to determine the fastest single-source, single-target shortest path algorithm for a given road network in the context of optimizing the response times of emergency responder units. We explore the viability of this solution by training several classifiers, including two variations of the Support Vector Machine (SVM) classifier, a standard decision tree, and a boosted decision tree on variations of Djikstra's shortest path algorithm and the A$\star$ algorithm with multiple heuristics. We evaluated the accuracy, recall, precision, and prediction time to determine the best classifier, and we found that our ultimate choice of XXXXXXX showed promise in most efficiently identifying the fastest shortest path algorithm to use, such that an appropriate path is output. These classifiers draw upon principles from Machine Learning and are relevant for our target application of optimizing emergency responder times.

1. Introduction

Past research efforts in the area of shortest path algorithms in the context of real-world road networks have focused on determining the best general shortest path algorithm. In a study by Zhan and Noon in 1998, a multitude of shortest path algorithms were evaluated on real-world road network data (**?**). The paper identified the PAPE graph growth algorithm and variations of Dijkstra as the best general algorithms. However, these results do not take into account the properties of specific graphs. 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 that works well most of the time.

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, as well as the A$\star$ shortest path algorithm.

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.

1.1. Motivation

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. Given relevant traffic conditions at any particular point in time, the weights of the input road network could be updated to reflect current conditions and provide the emergency response team with the current shortest path.

To achieve this goal, machine learning classifier trained to detect important features of a particular road network given source and target may be able to determine and implement the most efficient shortest path algorithm. In the context of our target application, this may save valuable time that could significantly improve the response times of emergency responders.

2. Data Set

Currently, there is no one universal structure to represent graphs, since often times different applications require different internal graph structures. We chose 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

University of Pennsylvania, CIS 419/519 Course Project.

our parser, and eventually to test our feature extraction. Unfortunately, our input graphs did not have positional data, so in the case of the $A\star$ algorithm, it was difficult to implement heuristics beyond the defaults defined in the *NetworkX* package.

The training and test datasets are extracted from the U.S. Census Bureau's Topologically Integrated Geographic Encoding and Referencing (TIGER) database, and were sourced from the Dryad Digital Repository under the title *Barrington-Leigh-Millard-Ball-PNAS2015-Century-of-sprawl-graphs-in-GML-format*. These data represent real-world street networks, and were output in GML format. Initially, we ran our classifiers on a test dataset of 100 graphs and achieved a maximum accuracy of around 75%, motivating an increase in the size of the dataset. However, we ran into memory errors in trying to parse the 200-graph dataset, leading us to move from batch features to online features.

3. Feature Generation

We used Python to preprocess our input data graphs before running them through our supervised learning algorithms. In preprocessing these graphs, we parsed the given 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 then passed this numpy array as the X argument in the fit method of our classifier. The features we computed are given below.

1. Average Clustering
2. Edge-to-Node Ratio
3. Maximum Edge Length
4. Radius
5. Eigenvector Centrality

We further considered Principal Component Analysis for dimensionality reduction, as the bottleneck in running time of this machine learning algorithm is in reading and choosing the features. However, given that we have only extracted a narrow range of features, this approach has limited usefulness.

4. Algorithms

4.1. Shortest Path Algorithms

For our shortest path algorithms, we chose the standard and bidirectional Djikstra's shortest path algorithms as well as the A$\star$ algorithm. All three algorithms are contained within the NetworkX (NX) package, a Python package designed to handle complex networks. We wanted our three algorithms for comparison to be uniform in their implementation, so we could reliably compare the running times for training and predicting our classifiers.

4.2. Learning Algorithms

We chose to sample four 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 believe that these classifiers may be particularly relevant for our purposes because they require 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. Once we determined that X was the most appropriate classifier, we tuned its parameters using GridSearch over the training set based on the true labels of the training set. Having chosen this classifier, we compared the prediction results of our classifier on our training set with the true labels for the training data. This comparison enabled 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.

4.3. SVM: Sigmoid and RBF Kernels

We used two types of support vector machines (SVMs) with both sigmoid and radial basis function (RBF) kernels. We chose to use support vector machines because we predict that there will be correlation between the graphs, and SVMs are capable of handling data that is not linearly separable. In the case that the final model may not be linear, an SVM with an RBF kernel is generally considered to be a default option. The sigmoid kernel, which can be used interchangeably with perceptron neural networks, has further been highly effective in practice. We predicted that the support vector machine classifiers would give us the highest accuracy of all of our classifiers.

4.4. Decision Tree

We chose to test a depth-3 decision tree classifier because it can handle the graph features we use as inputs to the base learning algorithm and can split on the shortest path algorithms relatively quickly. Further, a decision tree classifier has the unique benefit of being easy-to-use and user-friendly. An emergency responder would only need to input his starting point in order to use this classifier. However, we recognize that a downside of this approach is the

necessity of knowing the features of a graph in order to run this classifier. As a result of this analysis, we predicted that the decision tree classifier would be the fastest of all of our classifiers. We further tested an AdaBoosted version of the decision tree to improve accuracy and precision.

5. Experimental Results

We sampled four total variations of SVM and decision tree classifiers, and metrics for their learning accuracy precision, and recall can be found in the next section. For each classifier, we implemented GridSearch to determine the optimal parameters, which are again specified below. However, while GridSearch worked to optimize the test accuracy of each classifier, it resulted in a sacrifice in recall score in some cases, meaning that our classifier always predicted a single algorithm. As a result, in the case of the boosted decision tree and the SVM with a radial basis function kernel, we manually adjusted the parameters to find a classifier with the greatest consolidated accuracy, precision, and recall scores.

5.1. Error Rates

ROC curves demonstrate the ability of a classifier to effectively separate input data graphs by their respective most efficient algorithm. The higher the area under the ROC curve, the more effective the test result. Figure 1 below demonstrates the Receiver Operating Characteristic (ROC) curve given the GridSearch-chosen parameters for our SVM, decision tree, and boosted decision tree classifiers. As evident below, the SVM with an RBF kernel has the highest accuracy for GridSearch-chosen parameters.

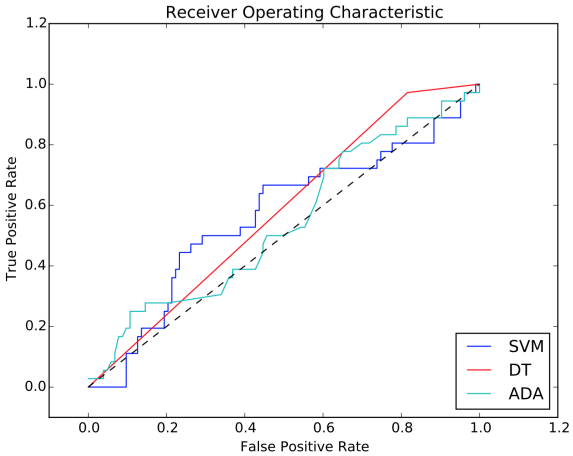
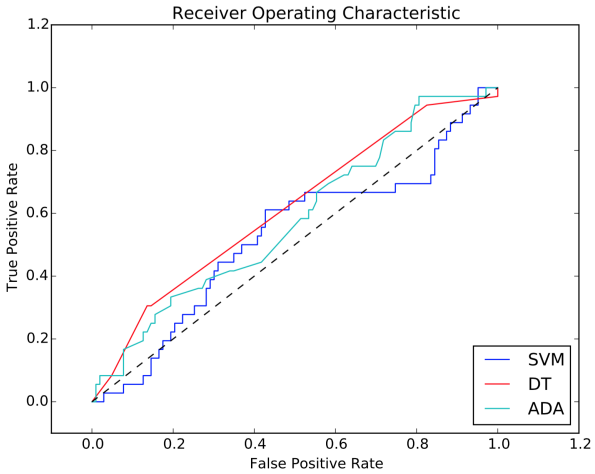


Figure 2 below represents the ROC curve given the manually optimized parameters for our SVM, decision tree, and boosted decision tree classifiers. Once these parameters were manually adjusted to improve test recall for the AdaBoost and SVM classifiers, it is clear that the depth-3 decision tree has the highest accuracy. However, the AdaBoost and SVM classifiers with manually adjusted parameters are viewed as more effective classifiers than their GridSearch-optimized counterparts because they are more discriminating in instances, as reflected in the their lower recall scores.



The full accuracy, precision, and recall metrics for each of our classifiers are given in Tables 1-3 below.

HI		HI	
	HI3	HI2	HI4

6. Algorithm Preference

7. Evaluation and Analysis

8. Conclusions

References

Blackwell, T.H. and Kaufman, J.S. Response time effectiveness: Comparison of response time and survival in urban emergency medical services system. *Academic Emergency Medicine*, 9(4):288–295, 2002.

Zhan, F.B. and Noon, C.E. Shortest path algorithms: An evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.