Toward Detecting Accidents with Already Available Passive Traffic Information

Robert W. Thomas and José M. Vidal
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC, USA

Abstract— **Traffic accidents occur every day, causing disruptions. The longer disruptions are in place, the more severe they may become as additional vehicles continue to enter the affected roadways. This paper looks at using passive data from a readily available source, smart phones, to detect traffic accidents automatically via machine learning algorithms and thereby allow additional alerts and actions to occur to minimize the disruption. Using simulated data, machine learning algorithms were scored for accuracy and the results were analyzed.**

Keywords—Multi-Agent System, smart cities, trace data

I. Introduction

Traffic accidents occur every day. Each and every one is disruptive to the flow of traffic. How disruptive is a product of the number of vehicles in the area, severity of the accident, and how long it takes to clear the accident. Efforts to clear an accident cannot begin until the accident is detected by appropriate resources such as police or a towing company. Furthermore, if the accident is undetected by vehicles in the area, drivers may unwittingly use affected routes, believing any initial congestion is simply due to traffic volume vice an obstruction. This only exacerbates the problem. There is an abundance of data that can be applied toward detecting traffic accidents and automatically disseminating disruption alerts. Today, smart phones and other GPS enabled devices are nearly ubiquitous in developed areas. This paper looks at how location information from individual smart phones of car drivers might be used to collectively detect traffic accidents. Specifically, how machine learning techniques might be applied to detect the presence of a traffic accident from passive traffic information of vehicles not involved in the incident. Once an accident has been detected, steps can be taken to notify interested parties, such as emergency responders who can resolve the incident and nearby vehicles that may want to avoid the area.

II. Background

Optimizing road traffic to reduce congestion has been a goal of many for some time. Research has gone into developing effective methods for relieving congestion including how to improve routing of traffic based on available network information [1] and increasing overall throughput of existing intersections [2].

Early detection of incidents is one way to reduce potential congestion. Machine learning has been used to this end in the context of analyzing video feeds at fixed intersections with the goal of detecting incidents [3][4]. The necessary infrastructure for monitoring roadways and intersections in this manner can be expensive though. This has led some researchers to use probe vehicles to gather data for use in qualitative analysis [13][14]. Methods reliant on Vehicular Ad-Hoc Network (VANET) enabled cars to relay data for analysis have also been explored [15]. These approaches show promise, but they also suffer from low volume of sensor data due to the dependency on the presence of probe cars and VANET equipped vehicles.

The presence of untapped data and how to make use of it is an area of consideration for Smart Cities [10]. There is a largely under-utilized data source that is already "deployed". Smart phones are everywhere, including inside vehicles on the road. Their sensors could potentially feed data to traffic management systems.

Expanding upon Waze's approach of capturing latent traffic data from smart phones to determine routing options [12], this paper looks at using that latent traffic data as input to machine learning algorithms tasked with distinguishing between normal congestion patterns and congestion caused by a traffic incident. The approach leverages a Multi-Agent System (MAS) to simulate traffic by modeling behavior of independent vehicles represented by agents traversing a roadway. Machine learning algorithms trained to recognize normal congestion versus congestion due to an incident for any given time step of a simulation are used to determine the presence of an incident for the monitored stretch of road.

A. NetLogo MAS Simulator

NetLogo is a programmable MAS simulation engine. It allows researchers to rapidly instantiate models to observe behavior of both individual agents and system collectives. It provides an intuitive user interface where one can add buttons and control widgets to easily manipulate a model to view different scenarios. An added benefit of NetLogo is that it is Java based so it can easily be moved across platforms. It is also open source with a number of tools and extensions available, including BehaviorSpace which allows for data collection from repeatable simulations runs specified by the user. The experiment described in section 3 below uses NetLogo version 5.2 [7].

B. *Machine Learning Algorithm Desciptions*

Machine learning algorithms are a broad category of algorithms that come in all shapes and sizes. The commonality that they all share is a means to refine their outputs based on previous input or results. The algorithms used in this study are supervised learning classifiers. For this category, training data with correct classification labels are provided. Each algorithm refines itself with the training data so that it can more accurately classify future examples of test data [5].

Three classifiers are used in this study. The classifiers used are Logistic Regression, a Bagging classifier of logistic regression, and an adaBoost classifier, as instantiated by scikit-learn libraries [6] included with Anaconda 2.3.0 [11]. Logistic regression is a linear model for classification which takes as input a feature vector [5]. The other two algorithms are ensemble methods which combine two or more sub-classifiers to form a committee that decides what class label input test data should receive.

Bagging classifiers augment a base classifier, in this case, logistic regression. Multiple versions of the logistic regression classifier are trained on different subsets of the training data to form a committee of sub-classifiers that ultimately judge the input test data. The idea behind bagging is that by varying the training data, errors spawned by over-correcting the model during training are marginalized, leading to a more accurate classifier [5][6].

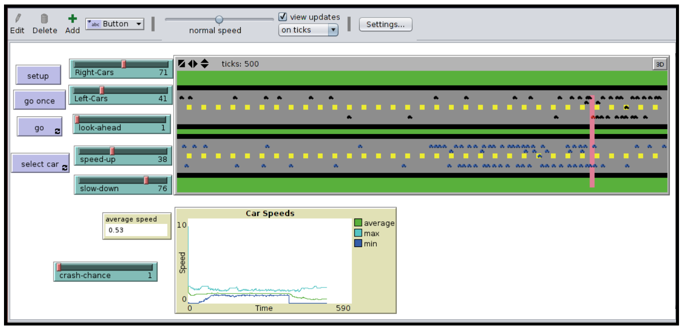
Adaptive boosting, or adaboost, follows a similar theory, but with a decision tree as the base classifier. A committee of sub-classifiers is formed from a decision tree. With each iteration of sub-classifier creation, training instances that were mislabeled are more heavily weighted to ensure those instances receive the more attention as successive sub-classifiers are spawned [5][6].

III. METHODOLOGY

The methodology used to explore how the machine learning algorithms perform at detecting accidents using passive traffic data was experimentation on simulated data. Simulation data sets were generated using NetLogo. The simulations used to produce both training data and test data to evaluate the effectiveness of different machine learning algorithms in correctly classifying observed information were produced using the same model. The NetLogo modeling and simulation engine was used to generate data sets which were in turn fed into machine learning classification algorithms, both for training and testing purposes, to determine the accuracy with which the presence of a traffic accident could be detected.

The model for the experiment was adapted from the NetLogo Traffic 2 Lanes Model [8] to simulate a monitored stretch of road. The model was adapted for 2-way traffic with two lanes going each way and to allow insertion of a non-reporting vehicle with a speed of zero. This crash vehicle creates an obstruction that vehicles traveling the same direction would

Fig. 1. Traffic 2x2 Lanes Model with Accident



Right-Cars 71
Left-Cars 41
look-ahead 1
speed-up 38
slow-down 76
average speed 0.53
crash-chance 1

have to navigate around. Vehicles traveling the opposite direction are impacted too. They face a distraction, causing them to slow down when passing the crash car. Fig. 1 illustrates the model in action with a traffic accident represented by the red car. The model allows for setting parameters for vehicle speed-up and slow-down rates and look-ahead which affects when a car begins speeding up or slowing down relative to cars in front of it. These settings work the same as in the original model. The crash-chance control sets the probability that an accident will spawn with each time an agent moves. The model only allows one crash per simulation. Once a crash occurs during a simulation run, it remains for the duration of that run. A foundational assumption is that everyone today carries a smart phone. Because of this, all vehicles other than the crash vehicle are capable of reporting passive traffic information.

Using the built-in NetLogo tool, BehaviorSpace, experiment parameters were set for light, moderate, and heavy traffic based on the total number of vehicles traversing the road in a given direction. Light traffic consisted of 25 cars, moderate consisted of 100 cars, and heavy consisted of 126 cars. Individual simulations then generated training data sets for each combination of traffic levels and corresponding test data sets with and without accidents. Other than the number of cars traveling in each direction and crash-chance, variables remained constant for each simulation run. Look ahead was set to one, speed-up was set to 38, and slow-down was set to 76. For training data, 20,200 training samples were generated by running 200 simulations of 101 steps each with crash-chance set to zero for half of them and 100% for the other half. 1,010 test samples were then produced from 10 simulations of 101 steps each with zero chance of a crash and 1,000 samples with 100% crash-chance. Ten samples were removed from the 100% crash-chance set due to a model limitation preventing a scenario from being initialized with a crash already in place at time step 0.

Test cases were broken into “no crash” and “all crash” sets to simplify analysis. Since the model assumes that the crash car cannot self-report, determining the presence of a crash relies on the behavior of the other cars in the system. A trivial solution would be to identify a crash scenario as that of one where some car reports a speed of zero based on it being impeded by the crash car. A simple classifier is introduced below to test for this condition. In real life, this can be problematic as heavy congestion leading to stop-and-go traffic also results in speeds of zero for one or more cars. Incorrect classifications on the no crash set represent false positives for incident detection. Misclassifying samples from the all crash set reflects false negatives.

Data output from BehaviorSpace running simulations against the traffic model consisted of Comma Separated Value (CSV) spreadsheets. Data recorded was speed, acceleration, x-coordinate, and y-coordinate of each reporting car at every time step. No allowance for noise or inaccurate reporting was made for this initial experiment. All cars were reporting with the exception of the crash car when present. The presence of the crash car was also recorded for use in classification training. Any time step beginning with the crash car present was reported as having a crash. Once the data was generated, it needed to be converted into a Support Vector Machine format able to be ingested by the scikit-learn machine learning instantiation used for the testing [6]. This was accomplished via the "convert.c" script which translates CSV data into SVM Light [9]. The first column of the SVM Light text file indicated crash classification and subsequent entries represented the four data fields reported for each car.

Once the converted data sets were ready, the classification algorithms needed to be instantiated and trained. Logistic regression is a standalone implementation, needing only to have the training data provided so it can fit its internal model to the data. The bagging classifier was instantiated with logistic regression as its base classifier. 10 sub-classifiers were used with the sample size and feature set parameters both set to 0.5, meaning sub-classifiers were trained with a random sample of half of the training data instances and half of the data points for each instance. The adaboost classifier was instantiated with 50 sub-classifiers, the default number, all based on decision tree classifiers.

Two additional classifiers were also instantiated. Voting classifiers were introduced by version 0.17 of the scikit library. They allow for ensemble classifiers to be formed from a heterogeneous mix of base classifier types as opposed to bagging or adaboost which use multiples of the same basic classifier type [6]. A simple majority vote classifier was formed using logistic regression, bagging, and adaboost classifiers as the ensemble members. These sub-classifiers were instantiated with the same parameters as their stand-alone counter parts described above.

The final classifier instantiated was a trivial case evaluator that used no training and simply declared a crash present if one or more cars reported a speed of zero. Short of a crashed car declaring itself, this is the most obvious indicator of an incident and therefore the threshold performance acceptable of any classification algorithm.

All four classifiers were fit to the training data applicable to each scenario prior to each experiment. That is, they were trained with the "heavy-heavy" training set simulating heavy traffic traveling both directions prior to being fed the uniform test data sets, all crash and no crash, for the heavy-heavy scenario. This process was repeated for all six traffic conditions to measure classifier results for each combination of traffic conditions. The classifiers were re-instantiated between scenarios so only the training data for the current scenario was considered for any given experiment.

IV. RESULTS

The results show promise. Overall, the most accurate classifier was adaboost, averaging over 85% accuracy across all traffic conditions compared with the trivial classifier performing at just under 68% accuracy. All of the machine learning classifiers outperformed the trivial classifier in terms of accuracy, though each also introduced false positives in doing so. While logistic regression barely performed better than the trivial case overall, the ensemble methods produced more accurate results by 10% or greater. The most influential factor affecting accuracy of predictions of the machine learning algorithms was the amount of traffic. The algorithms produced their most accurate results under the medium-light traffic scenario, followed by light-light scenario. In general, the heavy traffic scenarios were the toughest; however the variance between classifier accuracy was highest during the medium-medium with no crashes scenario. The ensemble methods, bagging and adaboost, performed relatively the same or better than logistic regression, as one would expect. The simple majority voting method was able to improve the results in some cases but usually reflected an average between the top two performers. The trivial case performed poorly at detecting crashes, heavily favoring a no crash verdict in all conditions. Though, it is not yet ready for production use as a stand-alone decision solution, at over 80% accurate for four out of six traffic conditions with adaboost, the method of using passive traffic information to identify traffic accidents is promising and certainly merits more research. Fig. 2 and Fig. 3 show the results.

Fig. 2. Overall Classification Accuracy

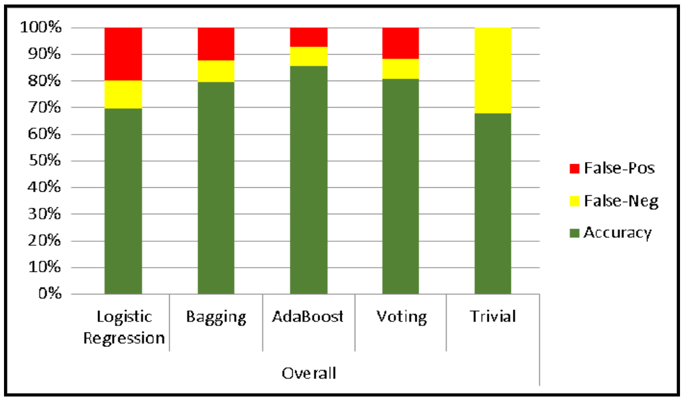
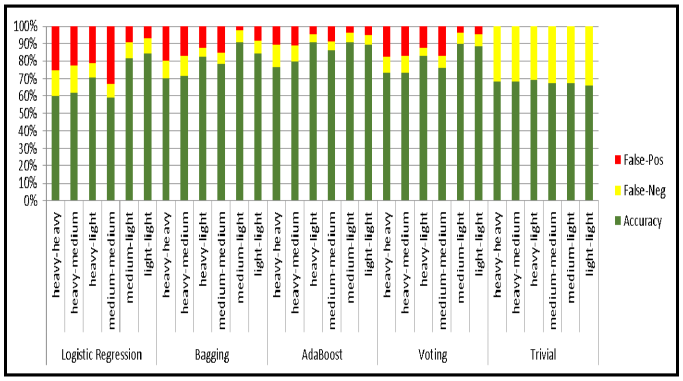


Fig. 3. Classification Accuracy by Traffic Condition



V. Conclusion and Future Work

As the results demonstrate, machine learning techniques can identify accident conditions from passive traffic information more times than not. The data is already out there as nearly everyone today carries a smart phone. All that is needed is an application to consume it and produce actionable knowledge. The approach presented is promising. It demonstrates there is knowledge to be gained from available trace data and that ensemble methods are better suited than single classifiers when dealing with highly variable dynamic environments.

When compared to similar work, the approach performed well. The machine learning algorithms' performance was comparable to the results in [14] and they out-performed the method used by [13]. The higher percentage of vehicles able to report was likely a significant advantage. The approach however requires more research and refinement to be useful in real-world conditions. Further, historical real-world data needs to be used to show the approach can transition successfully from the lab to production.

Future work will look at refining the data analysis to improve prediction accuracy while using real-world data of both traffic conditions and matching incident data. While historical data sets of this granularity were not publically available when these experiments were run, efforts are already underway to collect suitable data for use in future experiments. Individually, the classifying algorithms performed well, with the ensemble methods showing better results overall. Future efforts will look at expanding the ensemble approach to determine if a consensus between algorithms or approaches can improve accuracy.

References

[1] C. Gawron, "Simulation-Based Traffic Assignment," Ph.D. dissertation, Dept. Math, Köln Univ., Köln Germany, 1998.

[2] Dresner, Kurt, and Peter Stone. "Multiagent traffic management: A reservation-based intersection control mechanism." Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2. IEEE Computer Society, 2004.

[3] Ahmed, Tarem, Boris Oreshkin, and Mark Coates. "Machine learning approaches to network anomaly detection." Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques. USENIX Association, 2007.

[4] Kamijo, S., Matsushita, Y., Ikeuchi, K., & Sakauchi, M. "Traffic monitoring and accident detection at intersections." Intelligent Transportation Systems, IEEE Transactions on 1.2 (2000): 108-118.

[5] C. M. Bishop (2006). Pattern Recognition and Machine Learning. P. 3, 205, 655-657. Springer. ISBN 0-387-31073-8.

[6] (n.d.). Retrieved November 12, 2016, from scikit-learn: http://scikit-learn.org

[7] Wilensky, U. (1999). NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. http://ccl.northwestern.edu/netlogo/.

[8] Wilensky, U. (1998). NetLogo Traffic 2 Lanes model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. http://ccl.northwestern.edu/netlogo/models/Traffic2Lanes.

[9] Reagents of the University of California. "convert.c." Retrieved September 05, 2015. http://www.csie.ntu.edu.tw/~cjlin/libsvm/faqfiles/convert.c.

[10] CITIES, SMART. "Trace analysis and mining for smart cities: issues, methods, and applications." IEEE Communications Magazine 121 (2013).

[11] *Anaconda*. (n.d.). Retrieved November 12, 2016, from Continuum Analytics: http://continuum.io

[12] *Waze*. (n.d.). Retrieved November 12, 2016, from https://www.waze.com

[13] Asakura, Y., Kusakabe, T., Long, N. X., & Ushiki, T. (2015). Incident detection methods using probe vehicles with on-board GPS equipment. Transportation Research Procedia, 6, (pp. 17-27).

[14] Kinoshita, A., Takasu, A., & Adachi, J. (2015). Real-time traffic incident detection using a probabilistic topic model. Information Systems 54 , 169-188.

[15] Baiocchi, A., Cuomo, F., De Felice, M., & Fusco, G. (2015). Vehicular ad-hoc networks sampling protocols for traffic monitoring and incident detection in Intelligent Transportation Systems. Transportation Research Part C: Emerging Technologies, 56, 177-194.