Enabling Vehicular Data with Distributed Machine Learning

Cristian Chilipirea[a], Andreea Petre[a], Ciprian Dobre[a**],
Florin Pop[a], and Fatos Xhafa[b]

[a] *University Politehnica of Bucharest, Spl. Independentei 313, Bucharest, Romania*
[b] *Universitat Politecnica de Catalunya, Girona Salgado 1-3, 08034 Barcelona, Spain*

Abstract. Vehicular Data includes different facts and measurements made over a set of moving vehicles. Most of us use cars or public transportation for our work commute, daily routines and leisure. But, except of our destination, possible time of arrival and what is directly around us, we know very little about the traffic conditions in the city as a whole. Because all roads are connected in a vast network, events in other parts of town can and will directly affect us. The more we know about the traffic inside a city, the better decisions we can make. Vehicular measurements may contain a vast amount of information about the way our cities function. Information that can be used for more than improving our commute, it is indicative of other features of the city like the amount of pollution in different regions. All the information and knowledge we can extract, can be used to directly improve our life.

We live in a world where data is constantly generated and we store it and process it at an ever growing rate. Vehicular Data does not stray from this fact and is rapidly growing in size and complexity, with more and more ways to monitoring traffic, either from inside cars or from sensors placed on the road. Smartphones and in-car-computers are now common and they can produce a vast amount of data: it can identify a cars location, destination, current speed and even driving habits.

Machine learning is the perfect complement for Big Data, as large data sets can be rendered useless without methods to extract knowledge and information from them. Machine learning, currently a popular research topic, has a large number of algorithms design to achieve this task, of knowledge extraction. Most of these techniques and algorithms can be directly applied to Vehicular Data.

In this article we demonstrate how the use of a simple algorithm, k-Nearest Neighbors, can be used to extract valuable information from even a relatively small vehicular data set. Because of the vast size of most of our cities and the number of cars that are on their roads at any time of the day, standard machine learning systems do not manage to process data in a manner that would permit real time use of the extracted information. A solution to this problem is brought by distributed systems and cloud processing. By parallelizing and distributing machine learning algorithms we can use data at its highest potential and with little delay. Here, we

* *Corresponding author. Email: ciprian.dobre@cs.pub.ro

show how this can be achieved by distributing the k-Nearest Neighbors machine learning algorithm over MPI. We hope this would motivate the research into other combinations of merging machine learning algorithms with Vehicular Data sets.

Keywords: big data; machine learning; MPI; cloud systems; distributed processing

1 Introduction

Vehicular data consists of all measurements that are executed and generated on the cars that participate in traffic. It can represents a vast amount of information about the cities we live in. As a society, we have reached a point of very high mobility; we all travel long distances to get to our jobs, to our schools, to our leisure activities. All this movement represents the life of a city. We do have the individual tools to measure what we are doing, what our destination is and what is the fastest way to get to it, but when we look at large groups of people participating in traffic we do not really understand what is happening, what are the common flows or patterns that people follow, how we make use of our roads. This is how we can use Vehicular Data, to understand the dynamics of our cities and to try and improve them.

The types and the complexity of data are constantly growing. Until recently only external data generators were used, like sensors on the road that would be able to detect the amount of used capacity of the road at a given time. These are complex systems that are difficult to build as well as expensive. Now, with the growth in the number of wearable devices and vehicles with on board computers, data can be generated from inside the traffic flow and at a higher velocity then we were ever able to. A simple smartphone has numerous sensors such as GPS or accelerometer, that can be used to gather data about a car: "Where is it?", "What speed does it have?" and even "Where is it going?".

With the vast number of traffic participants and the large number of people that own a smartphone, or a car capable of sending real time information about itself, Vehicular Data is growing at a large pace. Public repositories with traffic data already are available from projects such as T-Drive [22], Cabspotting [13], and taxicabs [7]. The data being collected can lead to very large volumes, making the process of extracting information out of it rather complex. Popular processing methods rely on Machine Learning algorithms. Currently an important topic of research, such algorithms provide an efficient way to analyze large amounts of data. The complexity of processes that stand behind traffic flow is so large, that only data mining algorithms - from the domains of structure mining, graph mining, data streams, large-scale and temporal data mining - may bring efficient solutions for these problems.

There is not much surprise that the topic of efficient processing of vehicular data is today quite popular with scientists and practitioners alike. Competitions like the IEEE ICDM Contest [18] are designed to ask researchers to devise the best possible algorithms that tackle problems of traffic flow prediction, for the

purpose of intelligent driver navigation and improved city planning. Projects such as iDiary [14] develop advance algorithms to filter/derive/infer information out of this Big vehicular Data. However, even if the vast majority of work concentrate on designing the best solutions to deal with information extraction, little work has been done towards optimizing the data mining process itself.

In this article, our first contribution consists in showing how using a basic, simple Machine Learning algorithm, k-Nearest Neighbors, we are able to drastically improve the processing of information about traffic inside a city (Section 3). We next show how the ML algorithm can be distributed over multiple machines, and how it scales with the number of processors it uses (Section 4). We present in Section 5 a discussion about why we need to make this systems truly scalable and why we should allocate researching resources into these problems. Section 6 presents our conclusions.

2 Related work

The importance and variety in uses of Vehicular Data and the use of Machine Learning to optimize traffic, is demonstrated by various authors [25][23][21]. Pau and Tse [26] present a solution where vehicles are used to measure air parameters as well as urban traffic. The data generated is then used to better understand the pollution levels in cities such as Macao. Cars represent a big polluter and the more information we have about cars the better we can understand their impact on pollution. Fu et al [15] present an assessment of vehicular pollution using data about the estimated number of cars. This type of work can enormously benefit from accurate vehicle usage data that could be extracted from large vehicular data sets.

Safar [28] presents the use of vehicular data to answer kNN queries. The author tries to find the k nearest objects to a location on a map. In addition, vehicular data is used to improve the response time of these queries. We note this work because of the use of both vehicular data and the k-Nearest Neighbors (k-NN) algorithm (in fact, the work is among the first to make use of the k-NN algorithm in conjunction with vehicular data). However, the way in which they apply the later, and the purpose, are vastly different than our work.

Processing of vehicular data is commonly done with statistical techniques. Williams and Hoel [31] present a solution where the data is provided by a road control agency (the highway agency), and the processing is designed to extract weekly or seasonal patterns in the usage of the analyzed highways.

Old systems (see Hsieh et al [17] or Coifman et al [11]) use video feeds to do vehicle tracking and classification. Such systems can still be used as Vehicular data generators, but results show they fail to scale well and/or incur high/prohibitive costs for large-scale adoption (i.e., a lot of cameras need to be placed around the city, and processing video feeds requires powerful specialized equipment). Zhu et al [33] present a system that uses RFID tags on vehicles, and a city wide network is designed to gather vehicular data. Even though this system is less expensive and more scalable, it still requires large investments in the

network and the willingness of the tracked vehicles to install an RFID tag on their cars. Systems later evolved to transmit real time data from inside a vehicle. Chadil et al [9] present a system that uses a custom board with GPS and GPRS to enable vehicle localization and tracking.

Even though vehicular data is very noisy, current research is trying to improve its quality. Schubert et al [29] present a comparison between the models used to increase the accuracy of location data. With the high error rate of GPS receivers, and the importance of geospatial localization for vehicular data, this type of research is vital for the improvement of such data sets. Brakatsoulas et al [8] take a different approach and try to match GPS data to a street map, doing all the necessary corrections in the process.

The use of machine learning techniques over Vehicular Data has been done in works like [1], where this method is used to improve traffic signal control systems by making these systems truly adaptive. However the dataset used is considerably smaller, it covers only one intersection, while we are looking at city wide data sets.

Similar to our work, Sun et al [30] present the use of Bayesian networks to achieve traffic flow forecasting. Unlike our work, their test set is provided by the Traffic Management Bureau of Beijing and it is given in vehicles per hour, a metric that is not always available. Because of this less noisy data set and its size there was also no need to improve the execution time of their solution.

The need for scalability in the processing of vehicular data is also identified by Biem et al [6], where they use a small compute cluster to process GPS data from taxis and trucks. This data is processed as streams and it is used to create real-time speed maps over the city.

Zhang et al [32] present a way to distribute kNN Joins over MapReduce. Unlike our solution, their work concentrates on Joins on extremely large data sets, the main algorithm is however similar and they do obtain a speedup of 4 with 20 reducers. A different solution to the same problem of kNN Joins is presented by Lu et al [20]. The authors also chose to distribute the computation using the Hadoop MapReduce framework. Another article discussing machine learning algorithms over Hadoop is [16], where multiple machine learning algorithms are tested over the Hadoop framework. Similarly, authors in [19] present a similar solution, Distributed GraphLab, aimed for Cloud applications.

Reducing the execution time of k-nearest neighbors has also been done by parallelizing it using GPU cores [5]. Their solution can be used in combination with the solution we present in this paper to achieve even greater processing speed boosts.

3 Application of ML algorithms on traffic data

The traces we use were obtained from the CRAWDAD public repository [12], a community resource for archiving wireless data. The website contains a large number of traces that can be used in Mobile Ad-Hoc Networking and Vehicu-

Table 1: Trace characteristics.

	Roma	**San Francisco**
of cars	316	536
of data items	11.219.955	21.673.309
Start Date	17-May-2008	01-February-2014
End Date	10-June-2008	02-March-2014

lar Mobile Ad-Hoc Networking simulations. We selected two popular vehicular traces to run our experiments on:

- Roma trace [2];
- San Francisco epfl trac [27].

Table 1 includes the main characteristics of these two traces. San Francisco is the largest one, with double the number of data items and more than half the number of taxis being recorded. The traces themselves have also been taken six years apart, with the more recent one being the San Francisco trace.

Both trace contains location coordinates of taxis (in a format similar to *taxi_id*, *timestamp* and *location* (*latitude* and *longitude*). We chose these two data sets because they span over a moderately large time period, and they both include a large number of entries (an aspect particularly important to evaluate scalability aspects).

We believe that in any type of long time measurement regarding human activity one should be able to distinguish a day-night pattern. This pattern has a few main characteristics. First, it is an excellent way to validate the data, and a lack of such a pattern might indicate that something is wrong with the data generation/recording process or that the data measures something that is not directly affected by human behavior.

The data was formatted to take the following header: *time_of_day* (broken in 30 minute intervals); *day_of_week*; *speed*. The *id* field was not relevant for our experiments. We also removed all the data points where speed was lower than 0.5 km/h for a longer time period (to exclude parked vehicles from our measurements).

The newly formatted data was then processed using the *k-Nearest Neighbors* (k-NN) algorithm. The entire data set was used as a training set, and for evaluation we created a secondary set with 48 *time_of_day* intervals and null speed values. The predicted results, indicative of the k-NN model, are visible in Figures 1 and 2. In the figure we represented the mean speed computed for the various times of day (this is much lower than the maximum permitted speed, as there are always cars driving slowly, as expected in congested cities). Furthermore, the traces contain data about monitored taxis, which have an unusual driving behavior compared to normal cars. For example, they slow down when they want to search for a customer or for a building from which they received a request.

In the same traffic conditions we expect that normal cars would introduce a higher mean speed. But there are still a lot of traffic events that lower the mean speed: cars have to stop at cross-roads, cars stop to pick up other passengers, and cars slow down when the driver is searching for a location, emergency vehicles force drivers to slow down or stop to give priority. All these lower the mean speed of cars within a city.

For all processing and all graphs we used a k value of 500, and the k-Nearest Neighbor algorithm was set to calculate the mean speed over all the neighbor values.

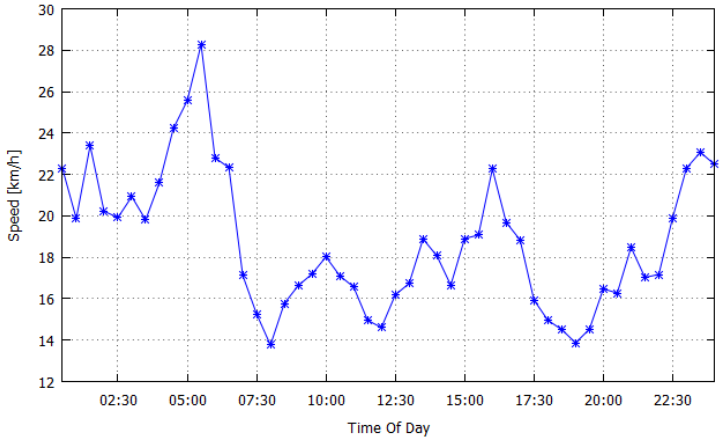

Fig. 1: k-Nearest Neighbors model for speed, Roma.

Figure 1 present the average traveling speed per time of day, obtained for the Roma trace. We can discern a day-night pattern in the obtained speed model. During the day, the speed lowers considerably compared to nighttime. This happens mainly because of high traffic during the day, with a lot of cars stuck in traffic – the mean speed is lowered for all the cars that participate in said traffic. This model is also a good predictor of what mean speeds to expect at a certain hour, or when it is best to make a trip so that you can achieve maximum speed. The figure is averaged for the entire city, but using the same principle, we can and did obtain data for different parts of the city, or even with fine granularity we can reach the street level.

The San Francisco model (Figure 2), shows a similar day-night pattern. This model also gave us some previously unexpected information. We are able to clearly identify the "rush hours", moments during the day where there are so many cars in traffic that the mean speed is lowered by a large factor. In the mode we can observe both the morning and the afternoon "rush hour" moments, where the mean speed is at its minimum. After a closer inspection of the Roma model, we identified the same drops in mean speed at similar time intervals.

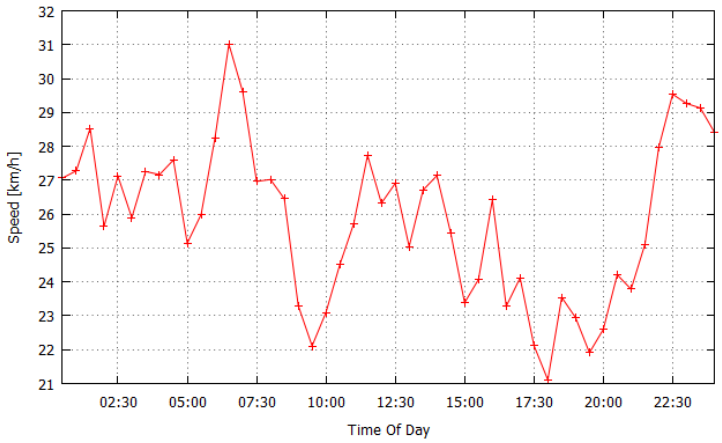

Fig. 2: k-Nearest Neighbors model for speed, San Francisco.

What we did not expect was that when we overlap the two models, we would find a relatively close correlation between them. In Figure 3 we plotted the overlapped data. Here, most of the raises and drops in the mean speed almost align. Considering the day-night cycle in mean speed is a sort of a "heart-beat" of the city, we find it extremely interesting that two very different cities in different parts of the world with different cultures, and data gathered years apart, have such a similar "heart-beat".

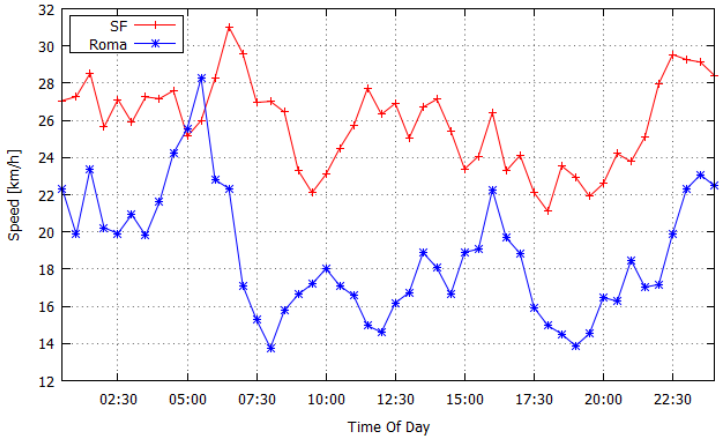

Fig. 3: k-NN San Francisco/Roma model comparison.

We note that the mean speed over the entire day in San Francisco is higher than the mean speed of the Roma trace. We believe this to caused because of geography; the city of Roma has hills, unlike San Francisco, and because it is an older city, dating back to the start of the Roman empire, the streets are not as wide, permitting less traffic, at lower speeds.

For the Roma trace we made another analysis. We wanted to extract density maps and see how these change as effect of the day-night difference.

Fig. 4: The city of Roma as obtained from Open Street Maps [24].

We divided the map into 10.000, or 100 by 100, sub-regions. Then, for each sub-region, we counted the number of cars between the hours 01:00 and 07:00 for the night, and between the hours 14:00 and 20:00 for the day, for each day individually. This data was then processed using the k-NN algorithm to extract a mean value for each sub region. The results can be seen in Figures 5 and 6. In these figures, a black color means a high number of cars, while a white color means a smaller number of cars, even 0.

It is very clear that during the day the car density is higher in the center of the city. By comparing these images with Figure 4, the map of the city, we can see how the cars follow the main city roads. We note that using this analysis we were able to map most of the roads in Roma using only GPS data from taxis.

Fig. 5: Car density during the night.

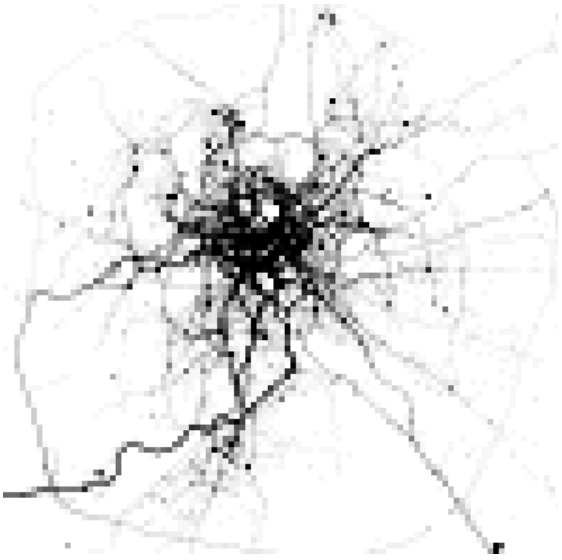

Fig. 6: Car density during the day.

4 Distributing k-nearest neighbors

Running k-Nearest Neighbors over multiple machines can be done in multiple ways. We chose to use the Massage Passing Interface (MPI) framework. MPI is

a distributed framework (and a message passing library interface speciiĭňĄcation for parallel programming) mostly used in high performance computing clusters.

MPI is appropriate for data-dependent iterative algorithms based on message passing. People have long opposed MPI to MapReduce (the next widest used model/framework for parallel/distributed ML algorithms – Chen et al [10] make an advanced comparison of the two), which is most suitable for non-iterative algorithms without lots of data exchanges. We chose MPI as an alternative to what we found in the literature about distributing k-NN using the Hadoop MapReduce framework (see Zhang et al [32], Lu et al [20], or Gillick et al [16]). To the best of our knowledge, today there is no MPI implementation of k-NN available.

Our solution reads both the training set and the test set in the first task ($rank == 0$), and this data is spread to all other tasks running on the distributed machines. Each machine then processes its part of the test set.

In k-Nearest Neighbors each element in the test set needs to be compared with all elements in the training set to determine: which are the k-nearest neighbors. After all the tasks are finished processing their part of the test set, the predicted value is calculated for each point in the data set and the data is gathered to the first task.

During our experiments we used different values for k, and we split the Roma test set into 80% for the training set and 20% for the test set.

The tests were run on a cluster of IBM Xeon E5630 rated at 2.53 GHz with 32 GB or RAM. During the experiments we used between 1 and 10 different such machines. We note here that each experiment was run 3 times and the results were very similar, in the graph we show the mean execution time between the 3 different runs.

In Figure 7 we can observe the execution time for the k-NN algorithm on the Roma data set. We used k values of 100, 300, 500, 700 and 1000.

The speedup for our implementation of the algorithm is almost linear with the number of compute nodes. This is shown in Figure 8, which shows the speedup for all the values of k we used in our experiment. For smaller values of k, the speedup is insignificant (this is visible for k=100, and degrades for lower values).

5 Discussion

Each experiment in this paper took between 30 minutes to 1 hour to execute on one single core machine, excluding data parsing or formatting. The traces contained data for just 1 month for over 500 cars. When we are looking at scaling these solution to city size we consider that every car is a potential data generator. This means millions of cars per city each generating one line of data every few seconds. To make things even worse, some machine learning algorithms, such as neural networks or convolution neural networks, have even higher execution times than the one we presented, but they could be used to extract even more interesting information from the data sets and to make more accurate predictions.

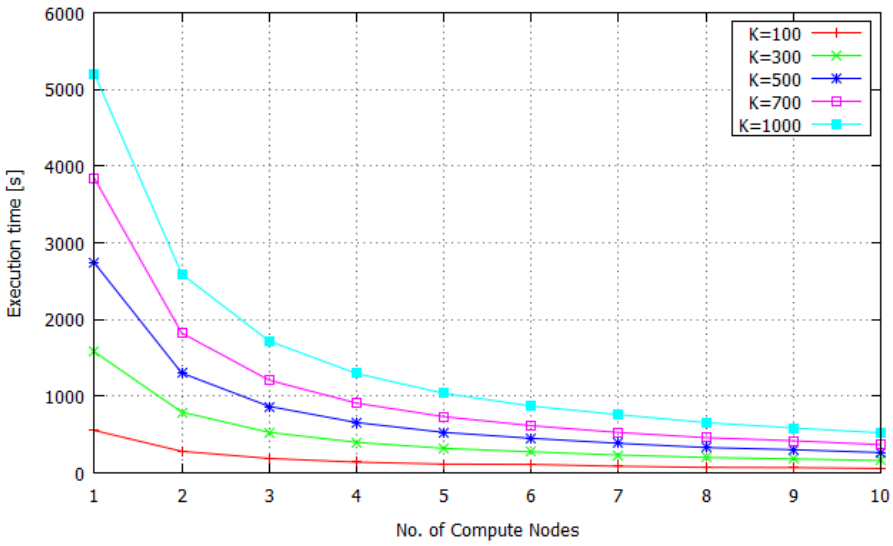

Fig. 7: Execution Time for distributed k-NN.

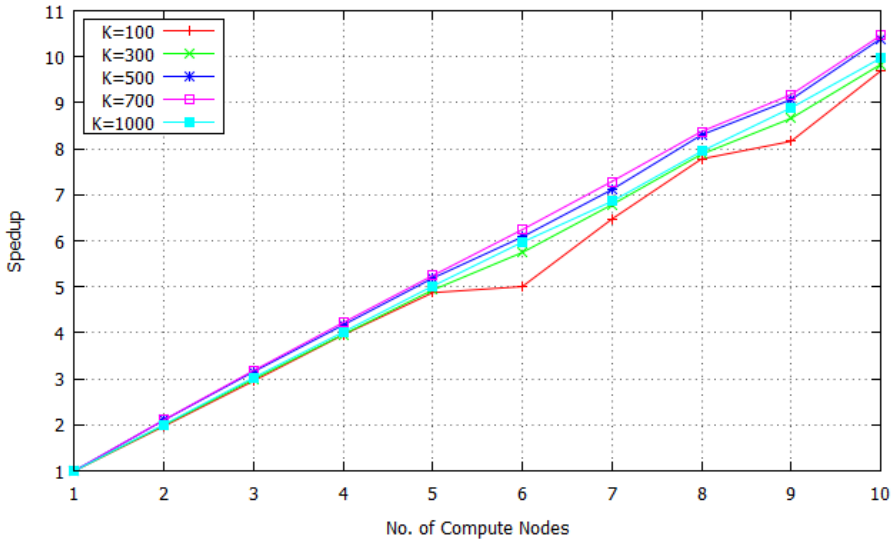

Fig. 8: Speedup for distributed k-NN.

With the extremely large number of data generators and the high execution time for machine learning algorithms building a complex real time solution can prove difficult. It is important to find the best ways to use all the resources provided by a modern computers, executing code on multiple cores, using both CPU and GPU, as well as enable the algorithms to make use of multiple machines.

Another factor that can raise the complexity of future Intelligent Transportation System (ITS) applications is the complexity of the generated data itself. Soon only timestamps and GPS coordinates will not be enough, and more complex sensors will provide a large variety of data. With the fast integration of smartphones that have powerful photo and video cameras, it is no longer far fetch to consider that soon one system will have to process video data generated by all the cars inside a city.

6 Conclusion

In this article we presented our analysis over two distinct vehicular traces, one for the city or Roma and another for San Francisco, using the k-Nearest Neighbor algorithm. We chose to use the k-Nearest Neighbors algorithm because it is one of the simplest (and yet powerful) machine learning tool.

With the analysis we showed that with the use of machine learning algorithms important information can be extracted from even basic vehicular data sets. The used data sets only contain time stamp and GPS coordinates (latitude and longitude) but other data sets may contain data from other sensors such as microphones, to measure noise levels or chemical sensors that could measure the level of pollution. With the minimal available data we manage to identify a day night cycle pattern, rush hours and we manage to build a model that could be used to predict high traffic intervals and regions with high vehicular density.

Then we showed how machine learning algorithms such as k-Nearest Neighbors can exploit the power of compute clusters or even clouds by reducing execution time through distributed processing. We achieve this using the popular MPI framework.

The need for highly scalable machine learning algorithms for use with vehicular data is further explored in our discussion section.

As future work we believe more machine learning algorithms should be tested in combination with vehicular data. We believe that with the right combination of data and algorithm surprising and interesting information could be extracted.

More machine learning algorithms should be parallelized or distributed so that they can be efficiently executed over multiple machines. A lot of work in this direction has been done in projects such as Mahout [3] or Spark MLLib [4], but not all machine learning algorithms have been integrated into these package. For instance *neither contain a k-Nearest Neighbors implementation.*

More vehicular traces need to be generated and analyzed. The vehicular traces we used only contained data from taxis, which made them slightly biased. A vehicular trace that contains all kinds of vehicles would be extremely inter-

esting for the research community. It would also be fascinating to have public traces that contain other sensor data such as pollution or noise levels.

Acknowledgment

This work was supported by the Romanian national project MobiWay, Project PN-II-PT-PCCA-2013-4-0321. The authors would like to thank reviewers for their constructive comments and valuable insights.

References

1. Abdulhai, B., Pringle, R., Karakoulas, G.J.: Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering 129(3), 278–285 (2003)
2. Amici, R., Bonola, M., Bracciale, L., Rabuffi, A., Loreti, P., Bianchi, G.: Performance assessment of an epidemic protocol in vanet using real traces. Procedia Computer Science 40, 92–99 (2014)
3. Apache: Mahout (2015), `https://mahout.apache.org/`
4. Apache: Spark mllib (2015), `https://spark.apache.org/mllib/`
5. Barrientos, R., Gómez, J., Tenllado, C., Prieto, M.: Heap based k-nearest neighbor search on gpus. In: Congreso Espanol de Informática (CEDI). pp. 559–566 (2010)
6. Biem, A., Bouillet, E., Feng, H., Ranganathan, A., Riabov, A., Verscheure, O., Koutsopoulos, H., Moran, C.: Ibm infosphere streams for scalable, real-time, intelligent transportation services. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. pp. 1093–1104. ACM (2010)
7. Bracciale, L., Bonola, M., Loreti, P., Bianchi, G., Amici, R., Rabuffi, A.: CRAWDAD data set roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/ (Jul 2014)
8. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proceedings of the 31st international conference on Very large data bases. pp. 853–864. VLDB Endowment (2005)
9. Chadil, N., Russameesawang, A., Keeratiwintakorn, P.: Real-time tracking management system using gps, gprs and google earth. In: Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTICON 2008. 5th International Conference on. vol. 1, pp. 393–396. IEEE (2008)
10. Chen, W.Y., Song, Y., Bai, H., Lin, C.J., Chang, E.Y.: Parallel spectral clustering in distributed systems. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33(3), 568–586 (2011)
11. Coifman, B., Beymer, D., McLauchlan, P., Malik, J.: A real-time computer vision system for vehicle tracking and traffic surveillance. Transportation Research Part C: Emerging Technologies 6(4), 271–288 (1998)
12. Dartmouth: Crowdad (2015), `http://crawdad.org/`
13. exploratorium: Cabspotting (2015), `http://cabspotting.org/index.html`
14. Feldman, D., Sugaya, A., Sung, C., Rus, D.: idiary: From gps signals to a text-searchable diary. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. p. 6. ACM (2013)
15. Fu, L., Hao, J., He, D., He, K., Li, P.: Assessment of vehicular pollution in china. Journal of the Air & Waste Management Association 51(5), 658–668 (2001)

16. Gillick, D., Faria, A., DeNero, J.: Mapreduce: Distributed computing for machine learning. Berkley, Dec 18 (2006)
17. Hsieh, J.W., Yu, S.H., Chen, Y.S., Hu, W.F.: Automatic traffic surveillance system for vehicle tracking and classification. Intelligent Transportation Systems, IEEE Transactions on 7(2), 175–187 (2006)
18. IEEE, TomTom: Ieee icdm contest: Tomtom traffic prediction for intelligent gps navigation (2010), http://tunedit.org/challenge/IEEE-ICDM-2010
19. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed graphlab: a framework for machine learning and data mining in the cloud. Proceedings of the VLDB Endowment 5(8), 716–727 (2012)
20. Lu, W., Shen, Y., Chen, S., Ooi, B.C.: Efficient processing of k nearest neighbor joins using mapreduce. Proceedings of the VLDB Endowment 5(10), 1016–1027 (2012)
21. Mavromoustakis, C.X., Kormentzas, G., Mastorakis, G., Bourdena, A., Pallis, E., Rodrigues, J.: Context-oriented opportunistic cloud offload processing for energy conservation in wireless devices. In: Globecom Workshops (GC Wkshps), 2014. pp. 24–30. IEEE (2014)
22. Microsoft, R.: T-drive: Driving directions based on taxi traces (2015), http://research.microsoft.com/en-us/projects/tdrive/
23. Mousicou, P., Mavromoustakis, C.X., Bourdena, A., Mastorakis, G., Pallis, E.: Performance evaluation of dynamic cloud resource migration based on temporal and capacity-aware policy for efficient resource sharing. In: Proceedings of the 2nd ACM workshop on High performance mobile opportunistic systems. pp. 59–66. ACM (2013)
24. OpenStreetMap: Openstreetmap (2015), https://www.openstreetmap.org
25. Papadakis, S.E., Stykas, V., Mastorakis, G., Mavromoustakis, C.X., et al.: A hyper-box approach using relational databases for large scale machine learning. In: Telecommunications and Multimedia (TEMU), 2014 International Conference on. pp. 69–73. IEEE (2014)
26. Pau, G., Tse, R.: Challenges and opportunities in immersive vehicular sensing: lessons from urban deployments. Signal Processing: Image Communication 27(8), 900–908 (2012)
27. Piórkowski, M., Sarafijanovic-Djukic, N., Grossglauser, M.: A parsimonious model of mobile partitioned networks with clustering. In: Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International. pp. 1–10. IEEE (2009)
28. Safar, M.: K nearest neighbor search in navigation systems. Mobile Information Systems 1(3), 207–224 (2005)
29. Schubert, R., Richter, E., Wanielik, G.: Comparison and evaluation of advanced motion models for vehicle tracking. In: Information Fusion, 2008 11th International Conference on. pp. 1–6. IEEE (2008)
30. Sun, S., Zhang, C., Yu, G.: A bayesian network approach to traffic flow forecasting. Intelligent Transportation Systems, IEEE Transactions on 7(1), 124–132 (2006)
31. Williams, B.M., Hoel, L.A.: Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. Journal of transportation engineering 129(6), 664–672 (2003)
32. Zhang, C., Li, F., Jestes, J.: Efficient parallel knn joins for large data in mapreduce. In: Proceedings of the 15th International Conference on Extending Database Technology. pp. 38–49. ACM (2012)

33. Zhu, H., Zhu, Y., Li, M., Ni, L.M.: Hero: Online real-time vehicle tracking in shanghai. In: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE. IEEE (2008)