

Analysing and Predicting Driver's Next Location

Author:
Sneha Gandhi-1457854

Supervisor:
Dr. Benjamin Graham

June 18, 2015

Abstract

Cutting-edge technology is being pioneered by researchers at Jaguar Land Rover to develop a truly intelligent self-learning vehicle that will offer a completely personalised driving experience and help prevent accidents by reducing driver distraction. This report examines the applications of machine learning techniques to analyse and predict driver behaviour. The method uses artificial neural networks which works on the person's driving history taking into account various other factors like day of the week, time of day, temperature and other weather information to make these predictions. All these various factors have been analysed to see their impact on predicting the next location. Taxi data of New York City has been used to study the possibilities, trying to reduce the negative log likelihood of training and testing dataset by including these factors, and seeing what training data should be used to obtain the least negative log likelihood for the test set by comparing the improvement in performance (decrease in negative log likelihood) after introducing different factors, one at a time and all together.

1.1 Introduction

Knowledge of a driver's destination is an important parameter for delivering useful information during the drive. For instance, an in-car navigation system could automatically show traffic jams, gas stations, restaurants, and other points of interest that the driver is expected to encounter along the way. If the navigation system can make an accurate guess about the general region to which the driver is heading, then it can intelligently filter the information it displays, reducing the cognitive load. While it would be possible to explicitly ask the driver about his or her destination, drivers would likely not bother to provide this information at the beginning of every trip. It would be much more convenient to automatically predict the destination. Predicting a driver's next driving location could be useful for giving the driver warnings about upcoming road situations, delivering anticipatory information, and allowing the vehicle to automatically adapt to expected operating conditions. This report presents a prediction method trained from a driver's past history and

other factors, which will be discussed in detail shortly. Specifically, I train an Artificial Neural Network to probabilistically predict the next driving location. Also, I try to reduce the negative log likelihood of training and testing data. This report next describes the previous literature in this field followed by a detailed description of the data used. It continues with a description of the factors used to make the predictions followed by a brief explanation of an Artificial Neural Network, after which I discuss how I approached the problem with some graphs showing the results.

1.2 Related Work

Here I introduce previous work in the field of next location prediction. Karbassi and Barth^[1] proposed a car-sharing application in the case of given driving start and end points, describe a route prediction and time-of-arrival estimation methodology that has been implemented with a real-world multiple-station car-sharing system. Most of the methods of vehicle route/destination prediction mainly depend on historical driving data to acquire possible routes/destinations in the future whereas my work also takes into account various other important factors like weather, time etc. For example, Krumm^[2] predicted a driver's near-term future path by a Markov model. Similarly, Simmons et al.^[3] adopted a hidden Markov model to predict destinations and routes, I, on the other hand have used ANN. Some research done at SAS Singapore^[4] uses Mobile Data Mining for Next Place Prediction- focuses on predicting the next location of a mobile user given data on his or her current location-a personalised mobile data mining framework - that not only uses spatial and temporal data but also other contextual data such as accelerometer, bluetooth and call/sms log, this is different from my work as I haven't taken into account person's cell phone details. Also, there has been some research to develop a fusion of machine learning algorithms to monitor the movement of mobile device users^[5] which is again different from my work as I haven't used the mobile data of the person.

2.1 The Data

It covers four years of taxi operations in New York City from 2010-2013 and includes 697,622,444 trips, taken from the University of Illinois website^[5]. The data is stored in CSV format, organized by year and month. In each file, each row represents a single taxi trip. Fig 1 below gives a screenshot of this data. The data is organized as follows:

- medallion, hack license, vender id, rate_code, store_and_fwd_flag
- pickup_datetime: start time of the trip, mm-dd-yyyy hh:mm:ss EDT.
- dropoff_datetime: end time of the trip, mm-dd-yyyy hh:mm:ss EDT.
- passenger_count, trip_time_in_secs, trip_distance
- pickup_longitude and pickup_latitude: GPS coordinates at the start of the trip.

- dropoff_longitude and dropoff_latitude: GPS coordinates at the end of the trip.

medallion	hack_license	vendor_id	rate_code	store_anc	pickup_datetime	dropoff_datetime	passenger	trip_time	trip_dist	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:06	6	360	1.24	-73.983803	40.743511	-73.990311	40.729755
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:08	1	480	0.9	-73.989128	40.756531	-73.990028	40.748409
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:07	1	420	1.08	-73.984596	40.749428	-73.984261	40.757778
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:04	1	240	2.15	-73.960228	40.762081	-73.940956	40.788502
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:05	1	300	0.71	-74.00518	40.71109	-74.009277	40.706196
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:10	5	600	3.24	-73.986351	40.747646	-73.971405	40.781967
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:07	1	420	2.14	-73.973297	40.763821	-73.984329	40.738922
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:09	5	540	2.45	-73.990471	40.739239	-73.966988	40.766586
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:09	1	540	1.38	-73.99868	40.732079	-73.987068	40.7393
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:07	1	420	1.59	-73.991997	40.770023	-73.965904	40.758991
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:16	1	960	3.68	-73.989471	40.730877	-73.963837	40.718933
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:09	2	540	2.4	-74.003448	40.736263	-73.993195	40.760094
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:12	1	720	3.06	-73.958595	40.769024	-73.98645	40.730366
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:09	2	540	1.94	-73.982475	40.771992	-73.983734	40.749668
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:03	1	180	0.77	-73.990334	40.730652	-73.989855	40.725163
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:10	1	600	2.8	-73.989685	40.747513	-73.955147	40.765736
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:06	1	360	1.01	-74.002243	40.73365	-74.000191	40.74295
2.01E+09	2.01E+09	VTS	2		01-12-2013 00:00	01-12-2013 00:35	1	2100	21.36	-73.782692	40.644493	-74.0149	40.716522
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:17	1	1020	2.29	-74.007645	40.740871	-73.989563	40.719013
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:06	1	360	1.34	0	0	0	0
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:17	1	1020	6.02	-73.991081	40.761238	-73.94915	40.827988
2.01E+09	2.01E+09	VTS	1		01-12-2013 00:00	01-12-2013 00:06	3	360	0.75	-73.98687	40.736549	-73.987854	40.744102

Fig 1- A screenshot of data

Of all these attributes, I have only used **pickup_datetime**, **pickup_longitude**, **pickup_latitude**, **dropoff_longitude** and **dropoff_latitude**. Also, I haven't used the whole dataset, but, parts of it, extracting nearly 10,000 entries at random from different months and different years (each month in one year could have up to 80,000 entries). Fig 2(a) here is the plot of pickup coordinates (longitude and latitude) on the map of New York City, fig 2(b) is an example of how these points can be separated into grid indices, in this figure, I have divided the entire map into a 30 X 30 grid, for instance and fig 2(c) represents the number of times any location has been visited (as pickup location). The colour of each block in the figure represents the count of visits in that area for a sample of 10,000 random points chosen from the Dec 2013 dataset.

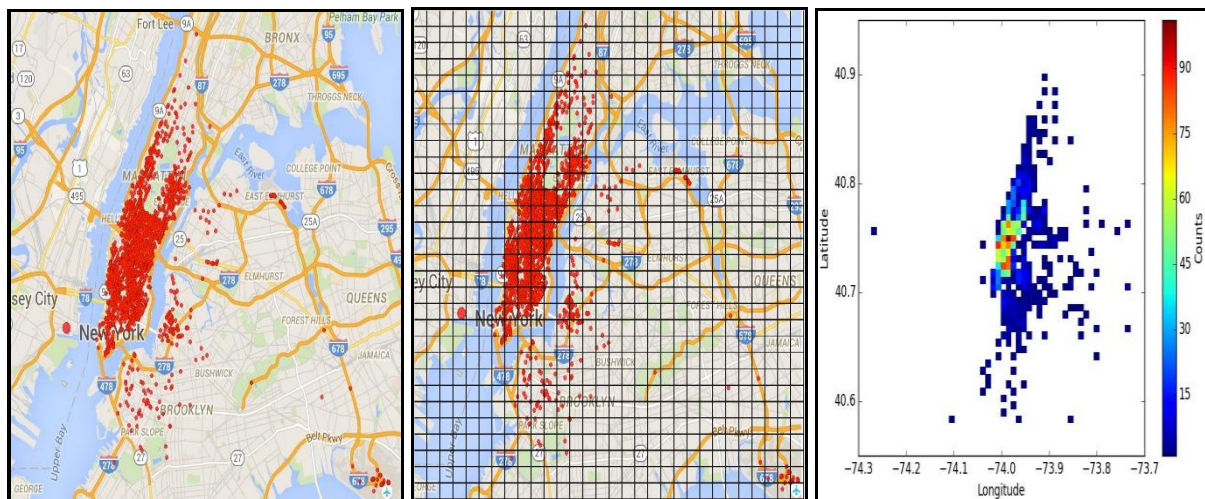


Fig 2(a) Pickup coordinates on map Fig 2(b) Map partitioned into grids Fig 2(c) Frequency of visiting a grid

2.2 Important Factors to Consider

- day of week
- time of day

- temperature and other weather conditions like wind, rain, snow and fog
- special event like music concerts (depending on persons interests)
- national holiday and other special days

Example scenario: A person is highly likely to go to work on a Monday morning or to go to drop his/her kids to school. If its weekend and sunny, the person might go for some outdoor sport which, on the other hand, is highly unlikely if it's raining. And say the person is a huge fan of music concerts, and never misses any concert in his neighbourhood, then he is quite likely to drive to the place on an evening when there is such an event in the neighbourhood. Similarly, the person is unlikely to go to work on a bank holiday.

2.3 Obtaining Information about Other Factors

I get the weather information from a website^[6] that has weather history of various cities stored as .csv files organised by month and year. The time of the day information is already there in pickup_datetime and information on whether it's a weekend or a weekday can be obtained from the date. I haven't used the information about some special event, as that would require a dataset with such information recorded, as deriving it (by looking at every trip that deviates from normal and then finding out if there was any special event at the time) can be difficult for a huge dataset.

2.4 Artificial Neural Networks

The concept of artificial neural networks comes from the biological network of neurons in brain. Brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called neurons. Human brain, for example, incorporates 10 billion neurons and 60 trillion interconnections. Fig 3(a) here depicts a biological network of neurons in brain. By using multiple neurons simultaneously, brain can perform its functions much faster than the fastest computers. In brain, information is stored and processed simultaneously throughout whole network, rather than at specific locations, thus, making both data and processing global rather than local. An artificial neural network consists of a pool of simple processing units communicating by sending signals to each other. These simple processing units are called neurons which are connected by weighted links passing signals from one to another. Output signal is transmitted through neurons' outgoing connection. Outputs split into number of branches transmitting the same signal and terminate at incoming connections of other neurons in network. Each neuron receives inputs from neighbours or external sources and uses them to compute output signal which is propagated to other neurons. Connections are weighted and weights are determined by training the ANN on data with known output values. The system is inherently parallel in the sense that neurons perform computations at the same time.

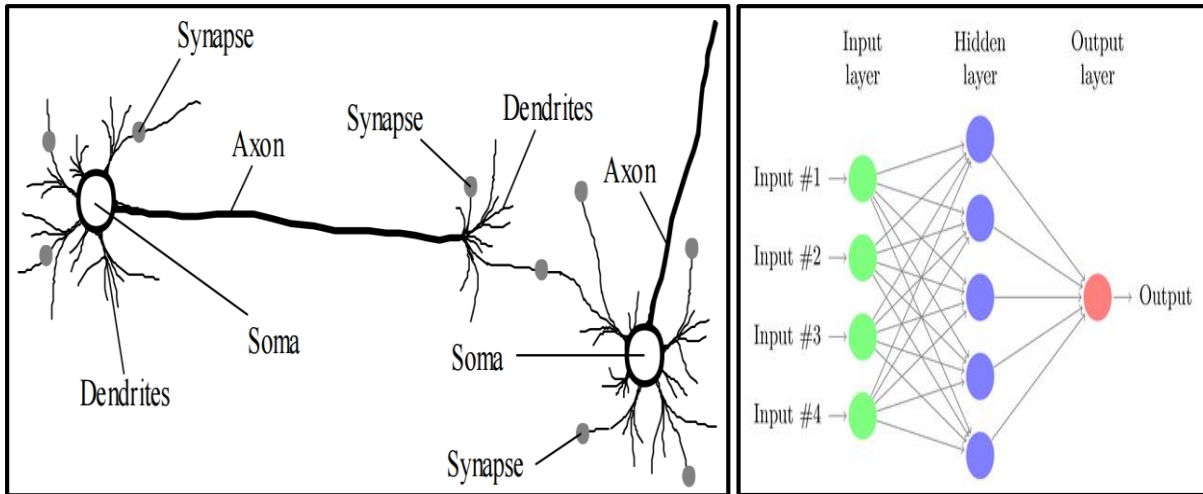


Fig 3(a) Biological network of neurons in brain

Fig 3(b) A 3-layer ANN

Fig 3(b) here represents a 3-layer artificial neural network with an input layer, a hidden layer and an output layer. Any node can receive input from its neighbouring nodes (see the directed edges) and, from some external sources. A node sends the same signal through all its outgoing edges, but the output received at the other end may be different for different nodes depending on the weight of the link connecting it to the node sending the signal. The most important reason to choose ANN is that they are highly robust, can work well even if training data contains errors, also, the prediction accuracy with ANN is generally high. ANN are also good in case we have some conflicting outputs (i.e., two instances with same input value have different outputs). ANN works well for discrete as well as real-valued input and the output too can be discrete or real valued. It also works when target value is unknown.

2.5 Negative Log Likelihood

The negative log likelihood value represents how close we are to making the right predictions. I have not considered error here, because, I am trying to see how close I get to making the right predictions and not how accurate these predictions might be. For example, if we have 50 possible output values, which can be 1 or 0. Suppose of these 50, the first 10 should be 1. And on testing, we get only first, say 7, as 1. This case will give us an error for this instance, whereas, the negative log likelihood value will be much lower than the case if we had received all the results wrong. Therefore, I have used negative log likelihood as my comparison parameter to see the effect of introducing various factors in the training and testing set. The closer the negative log likelihood value to zero the closer we are making the right predictions. If I have 30 possible values for output, then the maximum value of negative log likelihood can be $\ln(30) = 3.4012$. The lower this value gets, the better our prediction results.

3.1 Training and Testing Data- Approach 1

The computational substrate for my assessment of driving behaviour and destination predictions is a grid placed over the New York City, as shown in Fig 2(b). This grid is of different dimensions depending on the requirement in different case. Each cell is represented as an integer index, for example, for a 30 X 30 grid, the index for a pickup or dropoff location can be a number between 1 and 900. I convert the pickup and dropoff locations from a sequence

of (latitude, longitude) coordinates to the corresponding pickup and dropoff index. I have used the “neuralnet” package in r to obtain an artificial neural network shown in fig 4. For the predictions here, I divide the pickup longitude and latitude and dropoff longitude and latitude into a 30X30 grid, each grid number thus represents all the pickup and dropoff points in that grid. From these 900 grids, I consider only the most populous 51 grids. For one input instance, of these 51 points, one will be 1 and the rest will be 0.

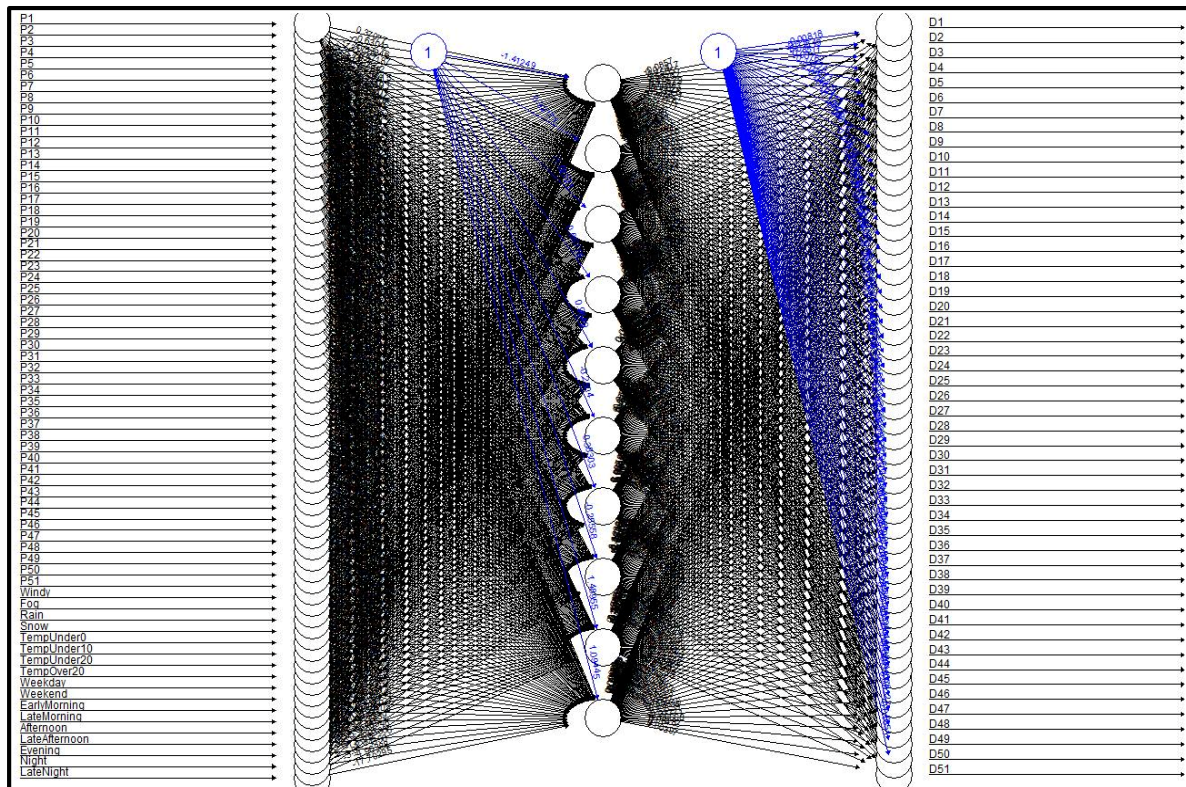


Fig 4 Artificial neural network trained and tested on nearly 2000 instances from Dec 2013

The neural network shown in fig 4 was trained on approximately 2000 inputs (with known output values, the known output being the dropoff location or the destination which will be 1 for only one of these 51 locations for any one instance). This neural network can then be tested on testing data from the same month/year or from different. It will give as output a value between -1 and 1 for all the 51 locations, the highest value (closer to 1) being the most likely next possible location. The problem with this method can be that it takes too long to train if I increase the number of training instance or the number of hidden layer nodes.

3.2 Training and Testing Data-Approach 2

This method is different from the previous method as here I will only get the output, which will be the index of one of the locations. This method is useful to see the impact of various factors and the reduction in negative log likelihood by introducing these variables. Here I will divide the input into x and y (longitude and latitude) and try to find out what might be the best to way get to the predictions. Using this method, I have done various analysis. Unless stated otherwise, my data will remain a reduced data frame, obtained by removing noisy data from the 10,000 instances chosen at random from Dec 2013 dataset.

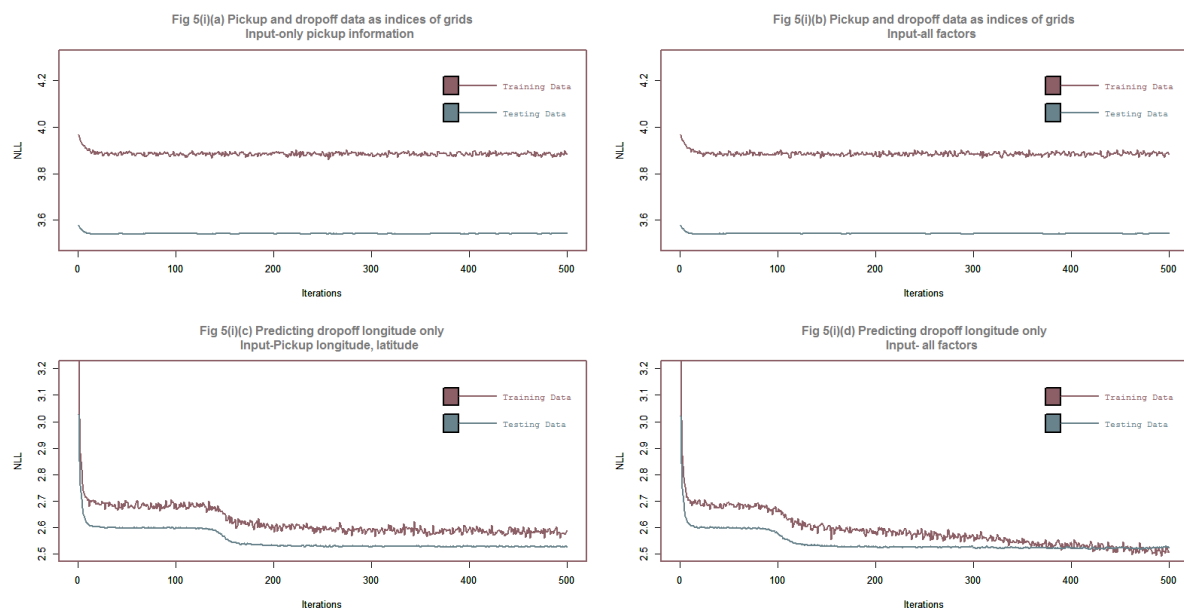
(i) Method of representation of pickup locations

In this comparison, we will see what method of representation of input is the best. By saying that one plot is better than the other, I mean that the drop in NLL is more or sooner, i.e., it takes less number of iterations to drop, or both. Fig 5(i)(a) and 5(i)(b) have been obtained by using the same 51 locations as in approach 1. We will shortly see why this may not be the best method.

For fig 5(i)(a) I have considered only the pickup information to make the prediction whereas in 5(i)(b) I add all other information as well. Notice how there is not any visible effect of adding additional information at all.

Consider now, fig 5(i)(c) and fig 5(i)(d), with predictions using pickup latitude and longitude only and all other factors as well respectively. Here, I take the longitude and latitude separately (in other words, try and see an approximate direction of movement by predicting one of the two at a time). So, now the output to be predicted is two parameters, predicted separately, with values ranging from 1 to 30. This method will obviously give a better NLL, as we are widening the possibilities here, as two outputs are to be predicted separately now. Notice, however, for the first method, $\ln(51)=3.932$ and for second, $\ln(30)=3.4011$, so, there is a huge difference between the x-axis of the first two and second two plots. But we can still safely say that the second plot is much better, as the drop is much higher and also there is a further improvement in this drop after adding information about other factors, it decreases more and sooner.

Fig 5(i)(e) and 5(i)(f) are plots for when the pickup points are considered as coordinates (longitudes and latitudes), without and with all other factors respectively. We see that this again is not the best method, comparing it with second one, this one has a less reduce in NLL compared to first one and also, it doesn't show any further improvement on introduction of other factors. Till now the best results we have seen is fig 5(i)(d), so, that will remain our point of comparison. From here on, we will be using the second method of representing pickup locations, unless stated otherwise.



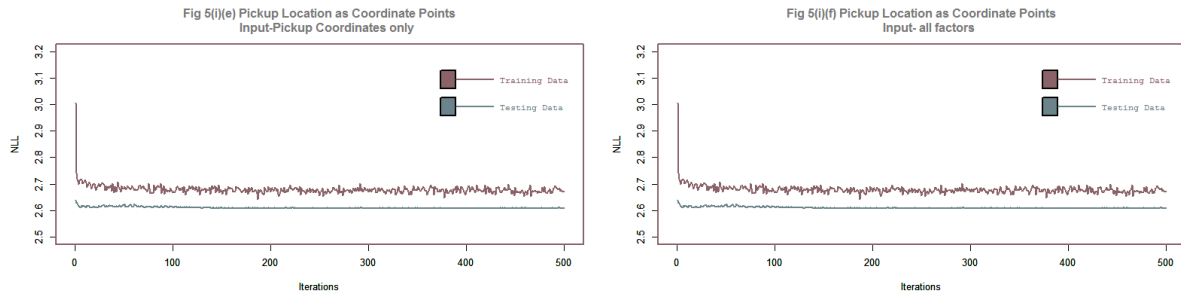


Fig 5(i) Plots comparing NLL for different representations of pickup locations (see title of each plot for details)

(ii) Effect of introducing different factors

Next, let's see the impact of introducing these parameters, how important is each parameter. Fig 5(ii)(a) and 5(ii)(b) are the plot of NLL when we add weather information only and when we add day of week information only, respectively. See how the results in these are better than those in 5(i)(c) but not as good as that in 5(i)(d) which is expected, as the results should improve with more information and fig 5(i)(d) shows the plot considering all the factors. Also, see the plot is better with weather information than it is with week of day information which implies that weather is a stronger parameter here than day of week.

Consider now figure 5(ii)(c) and 5(ii)(d), here, I have taken the training and testing data from the same month of different years, with all the factors included. When the training data is from Dec 2010 and the test data from 2013, the results seem to be getting worse, whereas, when training data is from 2012, it has good results. So, we see that if we move too back in time, the results tend to get a bit worse.

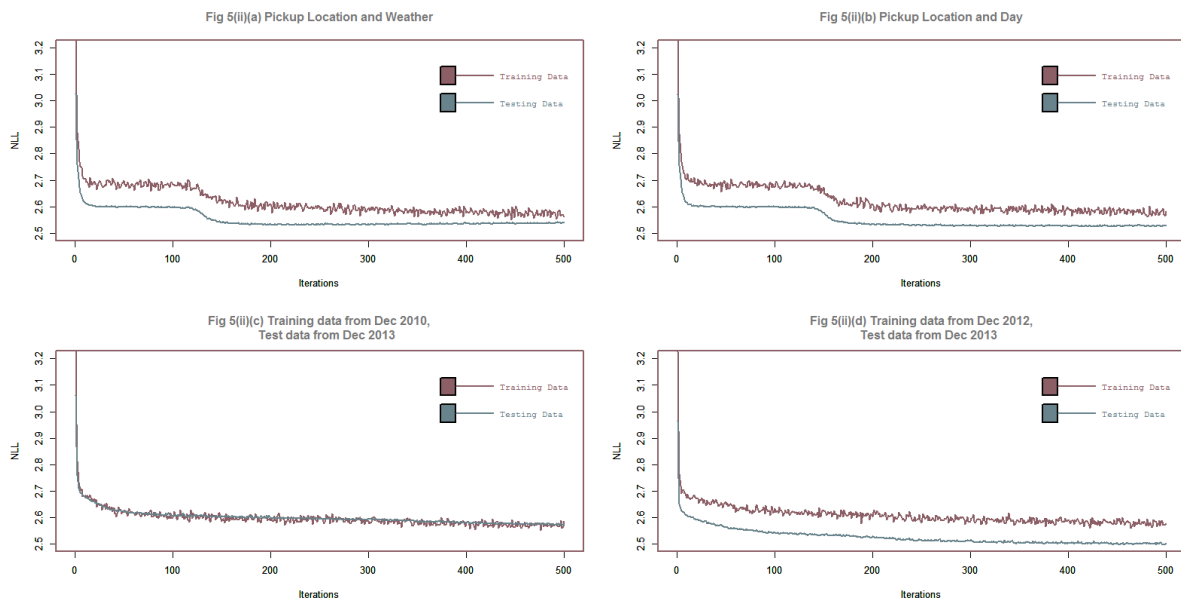


Fig 5(ii) (a),(b) Comparison of including different factors (c),(d) Testing and training data from different months (see plot titles for more details)

(iii) Month-weather and hidden layer nodes

Let's now see whether month is a representative of weather, or not. For this comparison in fig 5(iii)(a) and 5(iii)(b), I took equal number of random samples from all 12 months, year 2013. We don't see any difference in these plots, so, we can say that month might be equivalent to weather. Next, we will see the impact of the number of hidden layer nodes in fig 5(iii)(c) and 5(iii)(d) with 20 hidden layer nodes and 100 hidden layer nodes respectively. As would be expected, the results are clearly better when we take 100 hidden layer nodes.

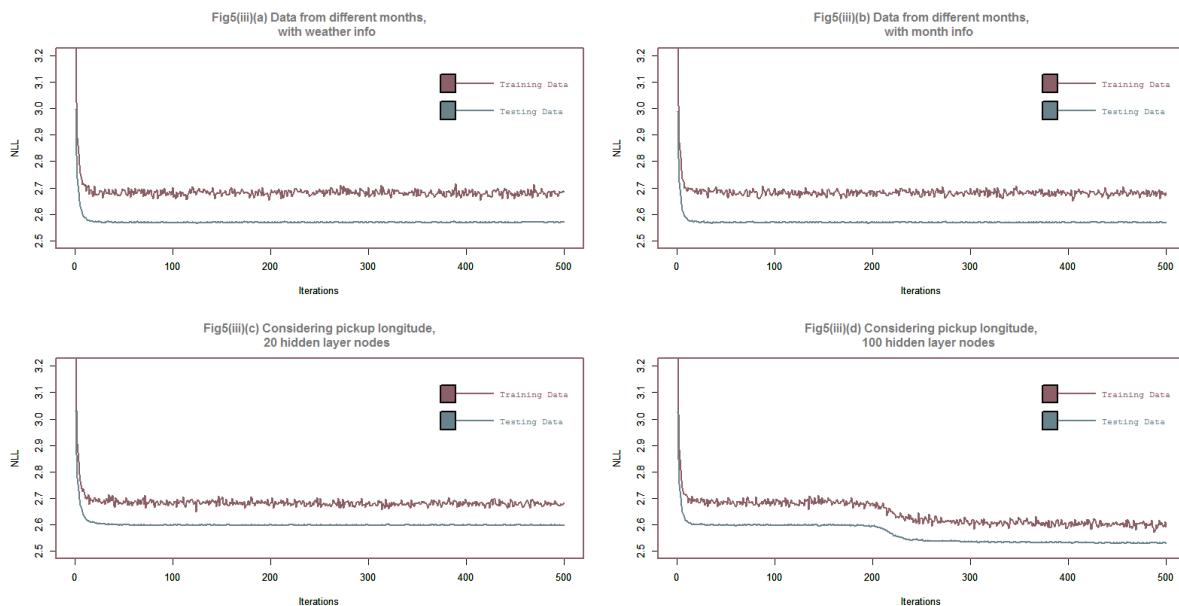


Fig 5(iii) (a),(b) Data from different months of a year, (c),(d) With different number of hidden layer of nodes

(iv) Analysis of day of week

Fig 5(iv) shows a violin plot of the pickup location during different times of day based on the distance travelled from the centre city coordinates: -74.0043, 40.4251. A violin plot is a method of plotting numeric data. It is a box plot with a rotated kernel density plot on each side. The violin plot is similar to box plots, except that they also show the probability density of the data at different values (in the simplest case this could be a histogram). From this plot, we can see that during late hours, the pickup locations are more concentrated around the centre of the city, whereas during rest of the day, it is more likely to go to some far off points too.

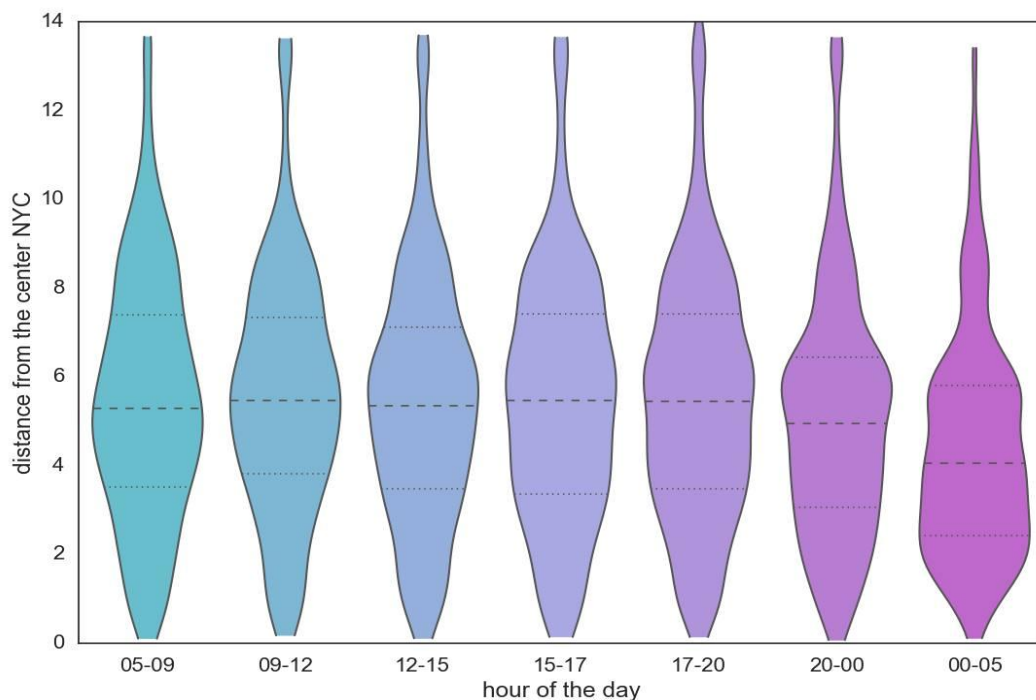


Fig 5(iv) Violin plot for the day of week

4. Challenges

- Taxi data may not be the best data to make predictions as it depends hugely on the person who might hail the taxi next and can, thus, not be personalised. These predictions could have been much better if we could use the data from a personal car. However, we could still see some detailed analysis of introducing various factors, thus, we can safely say that they will have a huge impact in helping predict the next driver location for a personal car.
- Possibly long training time with neural networks, especially in first case where I used the neuralnet package from R. It takes forever if I try to use more hidden layer nodes and a more number of training instances.
- It can be difficult to understand learned function (weights) obtained from the neural network.
- It might be hard to incorporate domain knowledge with neural networks.
- I could not include the information about any special events as it is hard to find information about special events dates with taxi data- different people might be going to different events and this information is not included in the dataset.

5. Conclusions

- We saw two ways of predicting the output using neural networks. From first method, we can obtain a range of possible outputs for any given instance whereas the second method is more useful to study the impact of various factors and compare various methods of predicting.

- This project can offer a completely personalized driving experience and help prevent accidents by reducing driver distraction.
- It can be trained to recognize who is in the car and then learn their specific preferences and driving style and then apply this learning by using a range of variables.
- Event history in the neighbourhood can be added, this will need more extensive training data.
- It can take over many of the daily driving 'chores', allowing the driver to concentrate on the road ahead.

6. Acknowledgements

I would like to thank my supervisor Dr. Benjamin Graham for providing ideas, and direction whenever I needed it most. Many thanks to the students and staff in the Centre of Complexity Science for their wonderful support.

This work was supported and funded by the Erasmus Mundus programme of the EU.

7. References

- [1] Byungun Yoon and Sungjoo Lee, *"An Operation Model of Carsharing Service: Application of Simulation Method"*, Journal of Traffic and Logistics Engineering, Vol, 1, No. 1 June 2013.
- [2] Krumm, J., "A Markov Model for Driver Turn Prediction", SAE Technical Paper 2008-01-0195, 2008
- [3] Ning Ye, Zhong-qin Wang, Reza Malekian, Ying-ya Zhang, and Ru-chuan Wang, *"A Method of Vehicle Route Prediction Based on Social Network Analysis"*, Journal of Sensors, Volume 2015 (2015), Article ID 210298.
- [4] Joao B´artolo Gomes, Clifton Phua, and Shonali Krishnaswamy, *"Where will you go? Mobile Data Mining for Next Place Prediction"*.
- [5] Cheng, C., R. Jain, and E.v.d. Berg, *"Location Prediction Algorithms for Mobile Wireless Systems"* Wireless Internet Handbook: Technologies, Standards, and Applications. 2003, CRC Press: Boca Raton, FL, USA. p. 245- 263.
- [6] The data- <https://uofi.app.box.com/NYctaxidata>
- [7] Weather history- <http://www.wunderground.com/history>