Deep Learning Based Origin Destination Matrix Prediction Through GPS DATA

AYOBAMI EPHRAIM ADEWALE*

University of Tartu
adewale@ut.ee

December 24, 2017

Abstract

Origin Destination matrix have been used over the years by transport agencies to understand and meet transportation demands but with the increase in world's population, previous method like Gravity Model, Statistical model and Equilibrium have become inefficient in their OD matrix predictions. The aim of this paper is to describe and take advantage of the recent advancement in deep learning to predict bus travel times by using OD matrix and Neural Network.

I. Introduction

One of the main transportation problem is meeting the demand for good transportation system.With the increase in the world's population, it has been difficult to fully understand and fulfill the demand of transportation. Over the years, demand of transportation have been known by presenting it through an Origin-Destination matrix. The Origin-Destination (O-D) matrix is a matrix with rows as origin and column as destinations, which is used to represent the number of trips or volume of travels from one origin to another destination. With an accurate OD matrix, we can understand and predict the traffic flow of a particular region or zone, draw road improvement plan, evaluate previous transportation investment performance and also predict the performance of a future investment. Previous OD Matrix have been drawn by making use of data obtained from link counts or transportation surveys and estimated through different techniques such as Gravity Model , Statistical Models, Equilibrium Models and Neural Networks.

In this thesis, we intend to adopt the concept of Origin Destination matrix and Deep learning techniques in the prediction of journey times in public transport. The resulting OD matrix will provide a detailed picture of journey times distribution in a region and can be used by transport agencies to plan transportation needs.

The remainder of this paper is organized as follows, section 2 talks about the literature review, Deep Learning Techniques in section 3, section 4 discusses the case study and the last section covers the result and conclusion.

II. Literature Review

In [1], Remya et.al took advantage of the computing abilities of Artificial Neural Network(ANN) which have been proven in various fields such as pattern recognition and fields related to prediction and estimation. ANN was used to develop a new technique that would

*keywords: OD matrix, Deep learning, Neural Networks, ITS

result into a more accurate O-D matrix estimation. The authors divided the OD matrix estimation problem into three, shortest path estimation, selection of Links and Training of the Neural network. Dijkstra algorithm was used for shortest path selection and the selection of appropriate links was based on some few assumptions and constraint. This was done because the availability of multiple links with unequal cost between any pair zones often affects the computation and accuracy of the OD matrix estimation. To minimize the deviation between the model outputs and target values, neural network Levenberg- Marquardt algorithm was used to train the data set and the performance was measured using Mean Squared Error(MSE). The result showed that the neural network model fits good in the analyzed scenarios but this were subject to several assumptions and constraints which might not give an accurate OD matrix estimate in a different scenario.

In the second paper [2] , Daehyon Kim and Yohan Chang adopted a type of ANN called multi-layer feed-forward network on a backward propagation model to solve the O-D estimation problem using link traffic counts. The result discussed in the paper showed that the backward propagation model provided a better estimation accuracy than the popularly adopted equilibrium-based O-D estimation. The authors mentioned that the model discussed is also suitable for real-time dynamic O-D estimation problem and guessed that it might be more reliable when applied to large complicated road networks with missing and noisy link data. The issues with the method discussed in this paper can be divided into two: first, the assumptions made based on the result of the model must be validated and verified by doing the same test on a large complicated road. Lastly, the result of a neural network model is more accurate when trained with large data set and there is a limit to the size of data that can be acquired through link traffic count. This means that a more accurate OD matrix estimate would have been obtained if the test made use of a huge GPS dataset.

In [3], Gusri Yaldi et.al took an interesting approach by testing the performance of three training algorithms of the NN models when used in generating an OD matrix estimating and the aim was to know the algorithm that would generate the most accurate OD matrix estimate. The NN model was trained with Backward Propagation algorithm, Variable learning Rate Algorithm and Levenberg Marquardt algorithm. The already trained network is then used to predict an OD-matrix of new data set, which has not been used in the training process. The experiment was carried out 30 times and the performance of the three algorithms was compared by looking at the Root Mean Square Error (RMSE) between the observed and the estimated trip numbers. The result of the experiment showed that the NN model trained with LM is better than the other two algorithms. The authors also mentioned that, there are other factors that can affect the performance of the NN model, such as the type of normalization method used. The experiment made use of a work trip data that is based on an home interview survey in Padang City, West Sumatra , Indonesia and it would be interesting to see the output when the test is done with a big data set such as an GPS bus data set because NN models gives better output when trained it big data set.

III. Neural Network Model

Neural Network is a concept that was inspired by the operation of the brain and so far, it has been successful in solving prediction and estimation problems because of their ability to find complex non-linear relationships. It makes use of Neurons known as nodes and signals are sent from one neuron to the other just like it happens inside the brain. Each neurons in the network processes the signal received from other neurons through some function known as the activation function and produces an output which is either forwarded to another neuron, displayed as final output or returned back into the network for further processing .

The neurons in the network are arranged in

layers and information or signals as it is called are exchanged between layers. The number of layers and the type of connection between them determines the architecture of the NN. For example, the figure 1 below is the smallest possible neuron network structure called the perceptron. It has a two layers, the input layer with two input neurons and the output layer with one neuron.

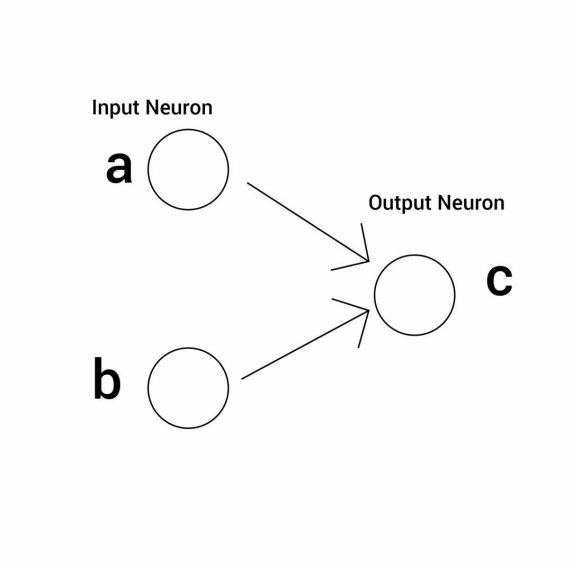

Figure 1: *Simple NN architecture called perceptron*

The input neurons read in the two features of the dataset into the network and the neuron in the output layer applies an activation function on the inputs. The output neuron c performs the simplest output function on a and b by multiplying the values with randomly chosen weights and added to a bias.

$$c = f(\sum_{n=1}^{2} w_n x_n + \theta)$$

$where \quad x_n \quad is \quad the \quad value \quad of \quad n^{th} \quad input$
$w_n \quad is \quad the \quad weigth \quad of \quad n^{th} \quad input$
$\theta \quad is \quad the \quad bias f \quad is \quad the \quad activation \quad function$

Perceptrons are very limited in what they can represent thus they are only use for representing linearly separable functions. When the complexity of the relationship between the input data and the output becomes complex, the number of layers in the Neural Network is increased. Figure 2 is a type of neural network called Multi-perceptron neural network with 2 layers, Input layer, Hidden layer and Output layer. This type of architecture is best at identifying patterns and trends in data for example in pattern recognition problems and time series problems.

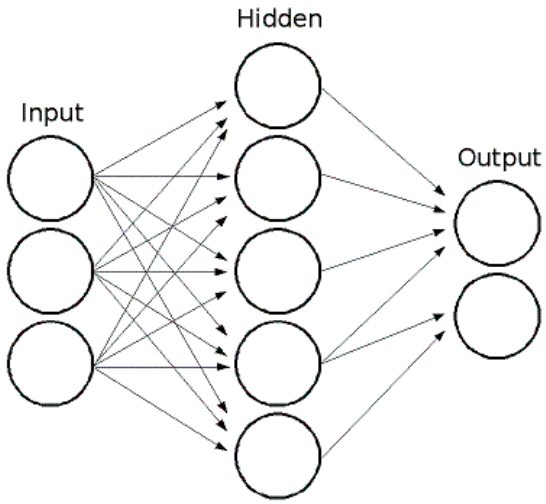

Figure 2: *Simple NN Architecture*

In this paper we focus on one type of multi-perceptron neural network called backward propagation which is a derivative of deep neural network. In this network, the weight for each input is initialized and the bias is added as discussed above. The computation of each layer is moved forward to the corresponding layers till its get the output layer. The result at the output layer is compared with the target and if the result differs from the target with a huge margin, the error is propagated back to previous layers in the network so as to adjust the previously used weights and bias.

The backward propagation algorithm tries to find the minimum and maximum of a function by iterating over the direction of the negative of the slope of the function to be minimized or maximized [7]. Mathematically, the error at the output layer is:

$$E = 1/2 \sum_{i=1}^{i+1} (O_i - t_i)^2 \quad MSE$$

and the algorithm tries to minimize:

$$\frac{\partial E}{\partial W_{ij}^{k}}$$

Using the gradient descent strategy, the error minimization problem becomes a chain rule of differentiation:

$$\frac{\partial E(n)}{\partial W_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ij}(n)} \quad [7]$$

After simplifying, the rule for updating the weights at each node becomes:

$$\Delta w_{ij} = \eta \delta_j(n) y_i(n)$$

Basically, the backward propagation is broken down into the following steps:

1. Feed forward computation into the network
2. Compute Error
3. Backward propagation to the output layer based on error
4. Backward propagation to the hidden layer
5. Adjust weight and repeat from 1.

Neural Network can be used to solve OD matrix prediction problem [3]. In this paper we try to model a NN architecture that can be used to predict OD matrix based on historical data.

While trying to solve an OD matrix prediction problem with NN , [3] compared different NN training algorithms Levenberg-Marqardt (LM), Quickprop and Variable learning rate (VLR) algorithms. The result showed that LM outperformed the other training algorithms and based on this, the NN model used in this paper was trained with LM algorithm.

The output of the model will be tested by computing the Root Mean Square Error (RMSE) between the expected value and the estimated value.

$$RMSE = \sqrt{\frac{\sum (t_{ij}^{t} - T_{ij}^{t})^2}{z}} [3]$$

Details	Value
Length	19km
Bus stops (One direction)	59
Journey Time (Excluding Peak Time)	60 minutes
Journey Id	2

Figure 3: *Details for Route 46A*

Where t_{ij}^{t} = the observed journey time from origin zone i to destination zone j for testing data.
T_{ij}^{t} = The estimated journey time from origin zone i to destination zone j for testing data.
z = the number of ij pairs.

IV. Data Collection

The dataset used for this study was obtained from the web page of Dublin City Councils traffic control and the data for the month of November was selected from the dataset. The route chosen was route 46A, it was selected because it is a busy route, it gives direct connectivity to both commercial and residential areas for commuters.

i. Data Description

In the dataset, each bus produces data every 20 seconds which is sent to the monitoring system. The data sent consist of the following:

1. The time-stamp of event, that is the time the event was sent to the monitoring system.
2. Current location of the bus in Latitude and Longitude
3. An identifier for the bus, term as the vehicle journey id
4. An identifier for the journey, term as the journey id
5. The journey pattern, which shows if the current journey is north bound or south bound.

6. Bus stop id , which relates the bus to a stop on its journey. In this dataset, every event sent by a bus has a bus stop identifier even when the bus is not currently at the bus.

ii. Data Cleaning and Preprocessing

Before fitting the data into the neural network model, i had to prepare the data for the model by cleaning the data and extracting the OD from the pre-processed dataset. The first step of preprocessing the data was to eliminate erroneous points or noise from the data. I grouped the dataset into journeys and point filtering was done on each unique journey.

1. The dataset for line 46A comes with four journey patterns 046A0001, 046A1001, 401001 and 400001. While analyzing the dataset , only 046A0001 and 046A1001 were frequently visited and they were also lengthy (covers upto 19km) . Based on this, journeys belonging to 401001 and 400001 were eliminated from the dataset.

2. Next, i eliminated all journey that had all its data points inside a 100m by 100m square. An example can be found in the figure below.

3. Next , i eliminated all journeys that had data points with large time jumps in it. Each bus are expected to send update about its location to the AVL every 20 seconds and it is possible to have delays due to obstruction etc. Thus, journeys that sent update to the AVL after 3 minutes were eliminate from the dataset.

4. Lastly, for each journey in the dataset, i marked all stops made by the bus at a different bus stops throughout the journey life span and journey with less than 8 marked stops were marked bad journeys and dropped from the dataset.

iii. Matrix Extraction

The OD matrix for each journey in the already filtered dataset was extracted by calculating the journey time between stops in the journey.

$$JT_{ab} = T_a - T_b$$

where JT is the journey time between stop A and B
T_a is the time stamp of the bus at STOP A
T_a is the time stamp of the bus after getting to STOP B

The table in figure 4 describes the OD matrix

Origin	A	B	.	.	.	Z
A	JT_{aa}	JT_{ab}	.	.	.	JT_{az}
B	0	JT_{bb}	.	.	.	JT_{bz}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
Z	0	0	.	.	.	JT_{zz}

Figure 4: *Origin Destination Matrix*

extracted from the dataset for each journeys in the dataset. In total, 1195 OD matrix was extracted from the dataset.

V. Data Analysis and Input Variable declaration

To determine the correlation between the journey times and other variables in the dataset, it was necessary to analyze the filtered data. By plotting the average journey time of trips in the dataset for both the north and south bound journey, some correlations were determined. From the plot in figure 5 and 6, we can see that the journey time is dependent on the hour of the day the journey started and the day of the week. It is also logical to know that the journey times also depends on the distance between the stops in the journey.

Therefore, from this analysis, the input variables considered for the model were distance between stops, time of the day, day of the week and the two considered stops. From this, the journey time was predicted. Since the day of the week and stops are categorical data, they were represented using One-Hot encoding.

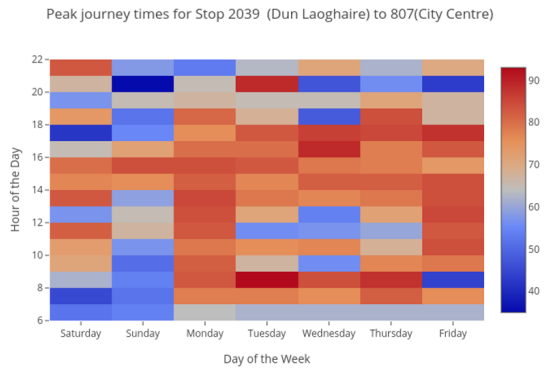

Figure 5: *North bound Journey*

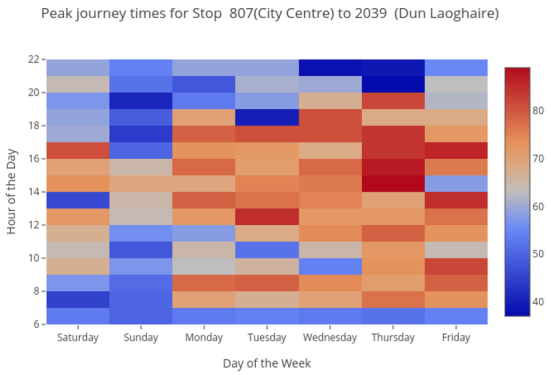

Figure 6: *South bound Journey*

VI. Result and Conclusion

As discussed in the previous sections, we construct a NN architecture which was trained using LM algorithm. The number of input neurons used in the NN architecture was 4 which is same as the number of independent variables in the dataset while the number of neurons used in output layer is 1 which is the dimension of travel time T to be predicted. To determine the number of hidden layers to use and the number of neurons in each layer, we conducted an experiment to obtain the best combinations of values to use and at the time of the writing of this paper, the best configuration can be found in the table in figure 7.

For the purpose of this seminar, the OD matrix obtained for the south bound journey was used as an input into the constructed NN

Details	Value
Input Layer	1 neuron
Hidden Layer	12 neuron
Hidden Layer	20 neuron
Output Layer	1 neuron

Figure 7: *NN configuration*

model. The data was divided into training and validation set, where 85% of the data was used as the training set while 15% was used as the validation set.

I analysed the result of the NN and divided my analysis into short and long journeys. Figure 8 below shows the graph of both the predicted and actual travel time for a single journey which was divided into nearest stops made by the bus before getting to its destination. The result showed that the prediction is always behind the actual travel time by an average of 1 minute. Figure 9 below shows the graph for

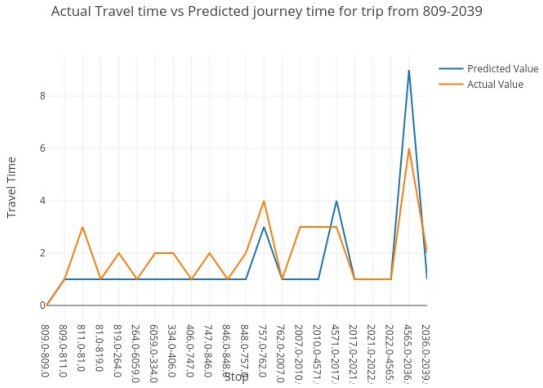

Figure 8: *Short origin-destination trips*

a single journey by analyzing different combinations of origin destination stops and we can see that the model makes quite poor prediction for long journeys.

From the two graphs, we can conclude that the model used is best for short trips while more improvement needs to be done for it to near accurate predictions for long trips.

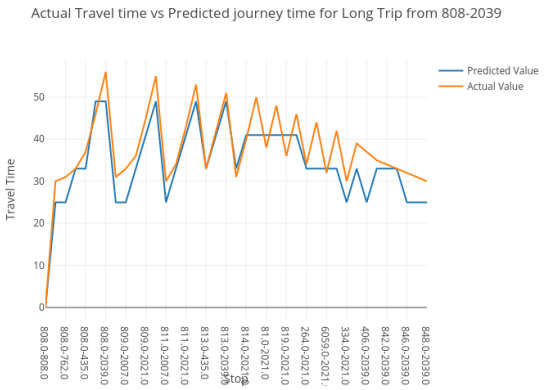

Figure 9: *Long origin-destination trips*

VII. Future Work

For future work , i will try to further denoise the data set and also try out some other neural network models like long short term memory networks (LSTM) which have been used to make accurate predictions for problems related to Time Series.

References

[1] Remya K P and Samson Mathew. "OD Matrix Estimation from Link Counts Using Artificial Neural Network (2013) " *International Journal of Scientific and Engineering Research*.

[2] Daehyon Kim and Yohan Chang (2011) " Neural Network-based O-D Matrix Estimation from Link Traffic Counts "

[3] Gusri Yaldi, Michael A P and Taylor Wen Long Yue " Forecasting origin-destination matrices by using neural network approach: A comparison of testing performance between back propagation, variable learning rate and levenberg-marquardt algorithms" `http://atrf.info/papers/2011/2011_Yaldi_Taylor_Yue.pdf`

[4] Jean DamascÃĺne Mazimpaka and Sabine Timpf"How They Move Reveals What Is Happening: Understanding the Dynamics of Big Events from Human Mobility Pattern " *International Journal of GeoInformation, January 2017*.

[5] Manojit Nandi "Density Based Clustering" `https://blog.dominodatalab.com/topology-and-density-based-clustering/`

[6] Jing Gao "Clustering: Density Based Method"`https://blog.dominodatalab.com/topology-and-density-based-clustering/` *Lecture Note for State University of New York College, Buffalo* .

[7] Abhishek Kar "Stock Prediction using Neural Network" *Department of Computer of Science and Engineering , IIT Kanpur* .