Link Travel Time Prediction using Origin Destination Matrix and Neural Network Model

AYOBAMI EPHRAIM ADEWALE*

University of Tartu
adewale@ut.ee

May 28, 2018

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

The birth of public bus transportation system has contributed immensely to the growth of cities in developed countries like New York in United State of America, Tallinn in Estonia, Beijing in China and Munich in Germany. It has been used to reduce traffic congestion, reduce the emission of greenhouse gas that are harmful to the environment, improve access to opportunities within connected cities, boost economy growth and lastly, improve quality of life in general. However, most people are still reluctant to take public bus and thus, prefer to go around with their private vehicles. This is quite understandable because the system is easily affected by weather, traffic signals, traffic fluctuations, peak hours and road incidents which often leads to delay in set schedules, irregularities in journey times and bus arrival times. In some countries, the difference between the set schedule and actual arrival times could be upto ten minutes or more. This does not only affect the plans of the commuters but also reflects on the economy growth. Based on this, there have been increasing demand for scientific techniques to solve the lingering problems.

In this article, we intend to adopt the concept of Origin Destination matrix and Deep learning techniques in the prediction of trip travel times in public transport. The resulting OD matrix will provide a detailed picture of trip travel 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
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 mainly because of its 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: Perceptron](image1.png)

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.

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

Where:
- $x_n$ is the value of $n^{th}$ input
- $w_n$ is the weight of $n^{th}$ input
- $\theta$ is the bias
- $f$ is the activation 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-layer 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: MLP Architecture](image2.png)

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:

\[
E = \frac{1}{2} \sum_{i=1}^{i+1} (O_i - t_i)^2 \quad MSE
\]

and the algorithm tries to minimize:

\[
\frac{\partial E}{\partial W_{ij}}
\]

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-Marquardt (LM), Quickprop and Variable learning rate (VLR) algorithms. The result showed that LM outperformed the other training algorithms. However, in this study, adaGrad algorithm is the algorithm of choice because it has been shown to work best for sparse dataset.

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

\[
RMSE = \sqrt{\sum_{i} (t_{ij} - T_{ij})^2 / z} \quad [3]
\]

Where \(t_{ij}\) = the observed journey time from origin zone i to destination zone j for testing data.

\(T_{ij}\) = 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

<table>
<thead>
<tr>
<th>Details</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>19km</td>
</tr>
<tr>
<td>Bus stops (One direction)</td>
<td>59</td>
</tr>
<tr>
<td>Journey Time (Excluding Peak Time)</td>
<td>60 minutes</td>
</tr>
<tr>
<td>Journey Id</td>
<td>2</td>
</tr>
</tbody>
</table>
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.

The calculated Travel Time and Journey Travel Time were both taken as the base line for the prediction model.

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
\( T_b \) is the time stamp of the bus after getting to STOP B

The table in figure 4 describes the OD matrix for each journeys in the dataset. In total, 1195 OD matrix was extracted from the dataset.

<table>
<thead>
<tr>
<th>Origin</th>
<th>A</th>
<th>B</th>
<th>. . .</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( JT_{aa} )</td>
<td>( JT_{ab} )</td>
<td>. . .</td>
<td>( JT_{az} )</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>( JT_{ba} )</td>
<td>. . .</td>
<td>( JT_{bz} )</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>. . .</td>
<td>( JT_{zz} )</td>
</tr>
</tbody>
</table>

Figure 4: Origin Destination Matrix

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 depend 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.

Furthermore, the travel time distribution for the south bound journey from Dun Laoghaire (Stop 2039) to Phoenix part (Stop 807) on weekdays was analyzed. Figure 7 shows the distribution and from the plot it can be seen that the travel times in the dataset follow a normal distribution curve. The left half of the plot represents the non-peak periods while the right half of the plot represents the travel time distribution at peak-periods.

It can also be deduced that the average travel time in the dataset is 80 minutes and with any particular trip starting at Stop 2039 to destination Stop 807, there is a 68% probability that it will be within 10 minutes from the average travel time.

VI. Preliminary Result

As discussed in the previous section, a neural network was modeled and adagrad is the learning algorithm of choice. The number of input neurons used is 6 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 these values are presented in table 1.

Furthermore, in order to improve the prediction accuracy of the model, the dataset was divided into two classes according to their estimated travel time before being used as an input to the neural network. Each class of the dataset is passed into a different neural network, however, the two networks have the same configuration highlighted in table 1. When prediction done, the result of the two network is then com-
Table 1: NN configuration

<table>
<thead>
<tr>
<th>Details</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>6 neuron</td>
</tr>
<tr>
<td>First hidden Layer</td>
<td>12 neuron</td>
</tr>
<tr>
<td>Second hidden Layer</td>
<td>50 neuron</td>
</tr>
<tr>
<td>Output Layer</td>
<td>1 neuron</td>
</tr>
<tr>
<td>Epoch</td>
<td>1000</td>
</tr>
<tr>
<td>Optimizer</td>
<td>adaGrad</td>
</tr>
</tbody>
</table>

Combined. Figure 8 gives a pictorial description of how this division was done.

![Figure 8: Dataset Class division](image)

### i. Model Evaluation

The performance of the model was evaluated in terms of accuracy by using Root Mean Square Error (RMSE). That is the predicted travel time (TT) was tested against the ground truth value which is the estimated travel time discussed in chapter three.

\[
RMSE = \sqrt{\frac{\sum (tt_{a,b}^j - TT_{a,b}^j)^2}{z}}
\]

Where \( tt_{a,b}^j \) = the observed travel time from stop \( a \) to stop \( b \) for a journey \( j \) in testing data.

\( TT_{a,b}^j \) = the ground truth travel time from stop \( a \) to stop \( b \) for a journey \( j \) in testing data.

\( z \) = the total number of links in the journey.

Figure 9 presents the behavior of the model when used to predict a single journey outside peak hours. It can be seen that the difference between the estimated travel time between two links is approximately equal to one minute. That is, in most cases, the model predicted accurately the travel time while in some cases, the difference between the predicted and estimated is less than one minute.

![Figure 9: Predicted link travel time for a journey outside peak hour](image)

In figure 10, the result of using the model to predict a journey inside a peak hour (16:00 is an example of rush hour period as shown in the analysis done in chapter 3) is shown. It can be seen that the model is not affected by peak hour as it almost accurately predicted the irregular travel time that might occur at that hour.

![Figure 10: Predicted link travel time for a journey inside peak hour](image)
In figure 11, the graph for RMSE observed for both peak hours and non-peak hours for a typical weekday is shown. The figure shows that the model performed better for peak hours compared to normal hours while on Friday’s, the RMSE error grows more than the other days.

Figure 11: RMSE error observed by day for Non peak and peak hour

For long jumps, Figure 12 presents the RMSE observed per day. As expected, the RMSE for long jumps are larger than that of short jumps, however, the prediction accuracy at 88% accuracy is still better than short jumps. The difference between the predicted and the ground truth is often between 3 and 4 minutes, which is acceptable given the type of data used and the factors considered. The model is also able to capture but non-peak and peak periods as shown from the plot. The error margin between peak hour and non-peak hours on Sunday for long trips is very low compared to that of short trips.

Figure 12: Long jumps measured RMSE

VII. Conclusion

The result showed that the NN model discussed is capable of making near accurate predictions when the number of stops between the origin and destination is at least 4 stops. However, if the travel time between two or three stops is between 1 and 3 minutes, the model gives poor prediction with prediction accuracy close to 70%. This is because the variability for short trips are often caused by time spent at traffic lights at intersections which are often unpredictable. To improve the expectations of the Model for short trips, the travel time predicted can be given in form of lower and upper bound pair.

Although, given a bigger dataset of two to three months, the NN model might be capable of making more accurate predictions.

References


[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


