Individual Project Report

Name : Chengyu Lou (oscar88feichang6@gmail.com)

Course : ENSE496AC Artificial Intelligence

Supervisor: Kin-Choong Yow

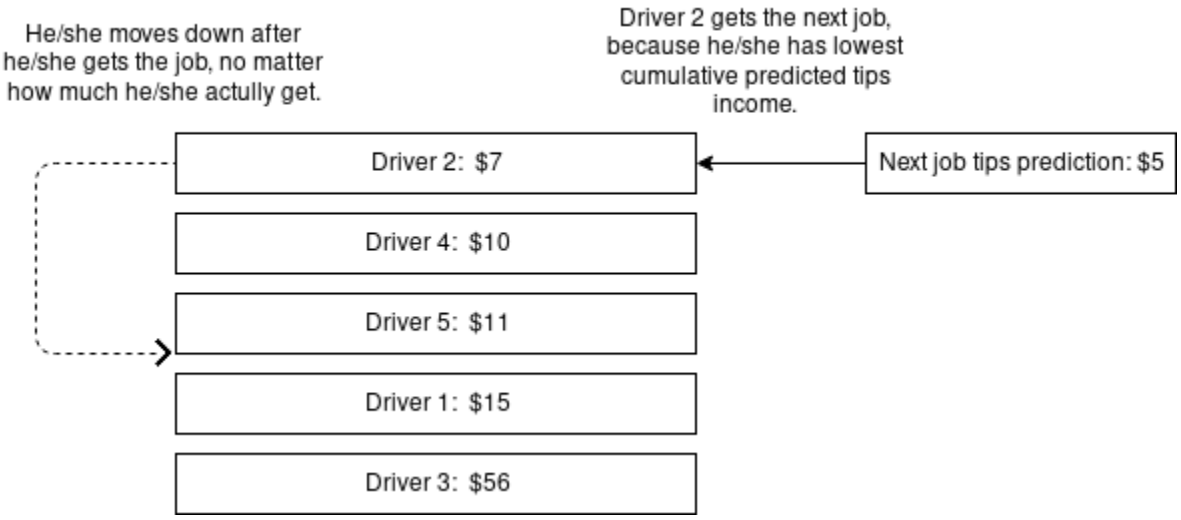
Title : Tips predictor for taxi drivers in Chicago

December 2019

Abstract

Over the last decade, the use of online booking method is gaining popularity in private commercial transport services such as Uber, DiDi, and Taxi.[1] Traditionally, Those companies use first in first out (Queue) method to give out jobs. That may lead to a non-even distribute of wages to drivers, specially in north America where tips are big trunk of the wages. A driver provides poor service may gain more wages than a well behaved driver because he/she is lucky. Also, it is dangerous for drivers to check their cellphone for best job offer while they were driving or pick up/ drop off people. Therefore, for the safety of people and fairness of drivers, a better job distribute system is needed.

This report describes the design and implementation of a tips predictor algorithm using artificial intelligence technique. The algorithm of this project focused on predict the tips by information given for each trip. The distribution system should include more methods in order to achieve a comprehensive solution. This could include a priority queue with smallest cumulative tips for each driver on top, therefore the driver with least opportunity to earn same amount of tips will get next job until the opportunities of getting tips are even out. However, this is not implemented for this project.



Introduction

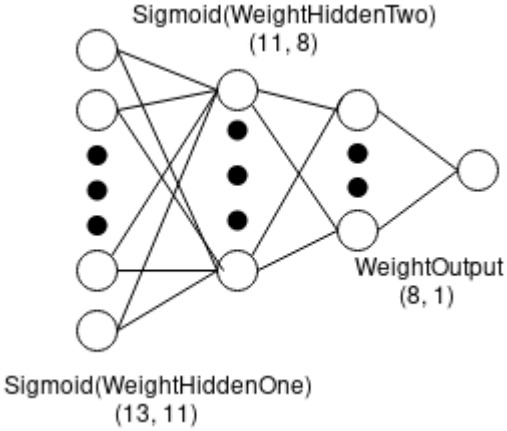
How to fairly distribute jobs is a hard task for the company, if they want to even out the opportunity of getting tips for each driver. The tips given are dependent on many factors, such as pick up location, how destination, and trip miles. For example, tips may higher if the driver pick up the customer from airport for extra luggage service. The ultimate goal for this project is to predict the tips by information given for each trip, so the company can evenly give out jobs to drivers at equal opportunity to get tips. Therefore, the tips income for drivers can be more dependent on their service.

Objectives

The objective of this project is to train a neural network using data obtained from Kaggle. The original data was collected by government of Chicago on year 2016. [2]

Design of the neural network

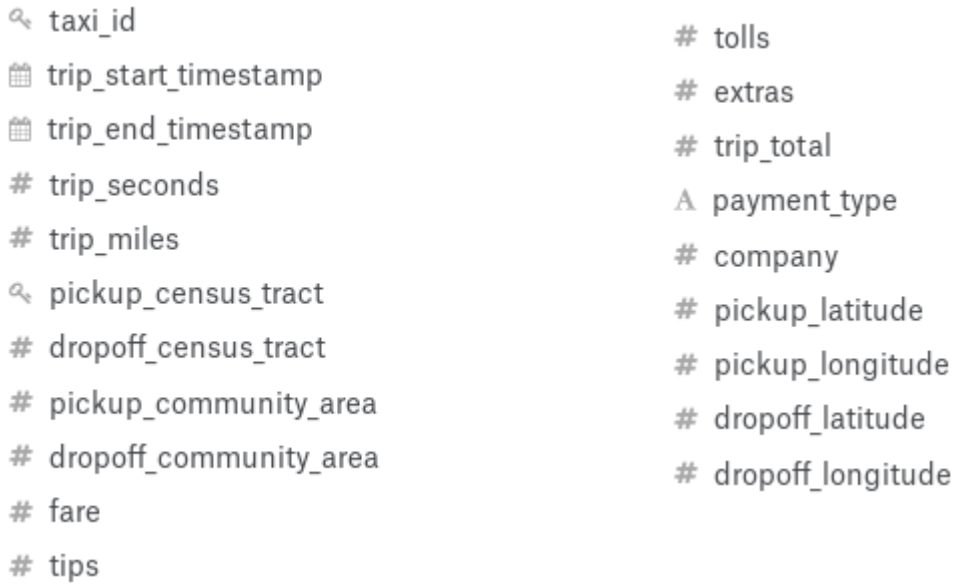
This Neural network consist two hidden layers, one input layer, and one output layer. We are using 13 attributes from data set as input and one attribute as output. First hidden layer has 11 neural, second hidden layer has 8 neural. Since we are using matrix form to do the computation, Hidden layer one is a (13, 11) matrix, second hidden layer is (11, 8) matrix, and the output layer is (8, 1) matrix.



The neural network has two Sigmoid layers for two hidden layers. The output layer does not have Sigmoid function, this way my output could be a positive real number. The biases for each layer are set to be 0.1 to reduce computation and to give empty columns some value to make it more accurate.

Data

The original data set has 20 attributes, they are as follows.[2]



However, many those attributes are dropped because they are largely empty. Also, I dropped taxi_id and payment_type for privacy reason. The attributes being used for this project are as follows.

trip_seconds	trip_miles	pickup_community_area	dropoff_community_area	fare	tips	
extras	trip_total	company	pickup_latitude	pickup_longitude	dropoff_latitude	dropoff_longitude

Data per-processing

I convert all empty “nan” values to 0, this is because “nan” values will give me undesirable result and 0 is more controllable for this project.

Also, I normalized data using sklearn library. This can prevent Sigmoid function gives me 1 for every column, since $1/(1+e^{-4})=0.98$ we do not want any value larger than four before apply Sigmoid function.

Formulas

The neural network using Sigmoid function to get active neural, because we want all inputs contribute to the network even with small percentage.

Sigmoid function:[3]

$$s(z)=\frac{1}{1+e^{-z}}$$

The derivative of Sigmoid function is:

$$s'(z)=s(z)\cdot(1-s(z))$$

Loss function[3]

Since this is a regression task, we are using half Mean Squared Error (MSE) for loss function

$$MSE=\frac{1}{2}\frac{1}{n}\sum_{i=0}^{n}(Y_i-\hat{Y}_i)^2$$

The derivative of MSE is:

$$MSEPrime=\frac{1}{n}\sum_{i=0}^{n}(Y_i-\hat{Y}_i)$$

Since we are using matrix form, the MSE prime would simply be

$$MSEPrime=(Y_i-\hat{Y}_i)$$

Forward propagation:

The formula for this forward propagation is as follows:

$$YHat=WeightOutput*Sigmond(WeightHiddenTwo*(sigmond(WeightHiddenOne*X)))$$

Since this project is using matrix to handle all calculation, the calculation are as follows:

number of rows for input X denoted as nX.

$$X(nX,13)\cdot WeightHiddenOne(13,11)=XOne(nX,11)$$

$$Sigmoid(XOne(nX,11))=ActivationLayerOne(nX,11)$$

$$ActivationLayerOne(nX,11)\cdot WeightHiddenTwo(11,8)=XTwo(nX,8)$$

$$Sigmoid(XTwo(nX,8))=ActivationLayerTwo(nX,8)$$

$$ActivationLayerTwo(nX,8)\cdot WeightOutput(8,1)=XOut(nX,1)$$

Back propagation

Back propagation algorithm can find how much each weight contributed to the overall error. This project is using half Mean Squared Error (MSE) as loss function to find the error. The derivatives of the loss function with respect to each weights are the cost derivative for weights we are looking for.

The derivative of loss function is as follows:

$$c'() = Yhat - Y$$

The formulas for the cost derivatives for weights are as follows:

$$\frac{\partial c}{\partial wo} = c'() * Sigmoid(W2 * Sigmoid(W1 * X))$$

$$\frac{\partial c}{\partial w2} = c'() * SigmoidPrime(W2 * Sigmoid(W1 * X)) * Wo * Sigmoid(W1 * X)$$

$$\frac{\partial c}{\partial w1} = c'() * SigmoidPrime(W2 * Sigmoid(W1 * X)) * Wo * SigmoidPrime(W1 * X) * W2 * X$$

To avoid recalculating the same function, the back propagation algorithm stores repeat variables in delta variables.

The implementation details are as follows:

```
delta3 = OutputError
djdwo = np.dot(self.ActivationLayerTwo.T, delta3)

delta2 = np.dot(delta3, self.WeightOutput.T)*self.sigmoid_prime(self.XTwo)
dJdW2 = np.dot(self.ActivationLayerOne.T, delta2)

delta1 = np.dot(delta2, self.WeightHiddenTwo.T)*self.sigmoid_prime(self.XOne)
dJdW1 = np.dot(X.T, delta1)
```

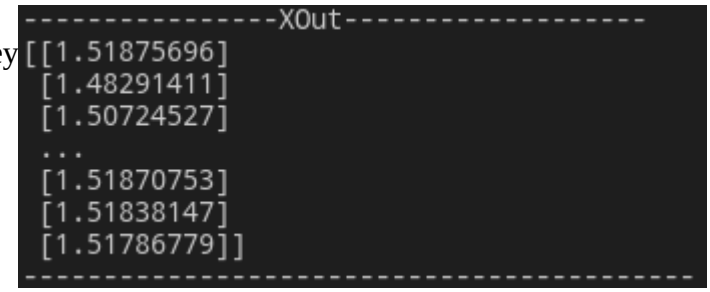
Learning rate

Initially I use 0.1 as my learning rate as many tutorials suggested. However, this lead to further away from lowest bottom of gradient descent, this is because in the tutorials they have small set of data, but there are around 1.68 million rows for every .csv file in the data set I am using. This escalates the gap between my Yhat and Y, because we are updating the sum of 1.68 million products times learning

rate to every weight variables. Therefore, I choose 0.00000001 as my learning rate to better leaning toward desired weight.

Result

After 50 iterations of training, the result shows that the output Yhat is more closer to the average of all tips given. This could means majority of the attributes are not strongly related to tips, or I should find a better to normalize the data. For example, the latitude and longitude are around huge numbers, after normalization they are either 0 for empty value or larger than 0.5 to 1. This could mislead the neural network to think this is more important than the fare.



Conclusion & Future work

This section summaries the overall conclusions for this project and discusses the potential future work. The key conclusions that can be made are :

- A full functioning neural network that help me to fully understand the technique.
- Reinventing the wheel helps me to have better control of the neural network. I can see the numbers after each layer, this helps me to find out how to normalize data and find a good learning rate.
- The predicted tips may not being useful, but this could also means randomly assign job to drivers is a fair way to equalize opportunity to get tips for drivers.
- The future work could be try different normalization methods and try different machine learning technologies or neural network structures.

Reference

[1]Taxi Market. (n.d.). Retrieved from https://www.mordorintelligence.com/industry-reports/taxi-market.

[2]Chicago, C. of. (2017, July 6). Chicago Taxi Rides 2016. Retrieved from https://www.kaggle.com/chicago/chicago-taxi-rides-2016#chicago_taxi_trips_2016_04.csv.

[3]Machine Learning Glossary¶. (n.d.). Retrieved from https://ml-cheatsheet.readthedocs.io/en/latest/index.html.

[4]Gradient Descent Derivation. (2014, March 4). Retrieved from https://mccormickml.com/2014/03/04/gradient-descent-derivation/.

[5]Backpropagation for a Linear Layer, Justin Johnson, April 19, 2017. Retrieved from http://cs231n.stanford.edu/handouts/linear-backprop.pdf

[6]Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. Cambridge, MA: MIT Press.

[7]Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. London: Springer.