Projet de Fin d'Études

Prediction of Vehicle turning behavior via Machine Learning

Xinran LIU
Polytech Tours, Département d'Aménagement et Environnement, 35 Allée Ferdinand de Lesseps, 37200 Tours

Abstract: In this paper, a combination of logistic regression and neural network is used to build a model to predict vehicle steering behavior based on vehicle driving data. The model was trained and tested with about 250,000 valid actual data, and the prediction success rate in retraining reached 92.01%, and in the random test, the prediction accuracy rate reached 92.02%. It shows that this model can well predict the turning intention of the vehicle through the driving data, and thus can assist the planning of the urban transportation network.

Key words: logistic regression, neural network, Machine Learning, urban planning, transportation

1 Introduction

Vehicle turning behavior is a very important part of traffic flow simulation. In the field of autonomous driving, the prediction of vehicle turning behavior is usually based on the vehicle trajectory during the steering process[1].In the field of vehicle engineering, the prediction of vehicle turning behavior is usually based on mechanical stress and vehicle design[2].However, in the field of urban planning, there is relatively little research on predicting turning behavior of vehicles. This article will use machine learning methods, via logistic regression models and neural network models, to predict the turning behavior of the vehicle to assist in forecasting the demand of the road network.

The parameters of the vehicle during the driving process can be divided into two categories, one is a quantitative parameter, such as speed, acceleration, and distance to the front vehicle, etc., and the other is a qualitative parameter, such as the driver's steering intention. These quantitative and qualitative parameters are related, but there is no obvious functional relationship. So, we can't quantify them through a simple formula. The self-learning advantages of machine learning can just solve this problem. It can mine information other than the mathematical relationship between the data itself, or potential information in unrelated data.

This article will build a machine learning model between the quantitative parameters of the vehicle and the vehicle's turning behavior, train the model with actual data, and finally test and analyze the training results.

2 Method

Machine learning models have become popular due to their ability to capture non-linear relationships between variables, making them capable of predicting high quality in multiple domains. Machine learning models can be broadly divided into unsupervised learning and supervised learning[3]. Cause the Prediction of Vehicle turning behavior is a kind of classification and we know the target of prediction (Go straight, turn right or turn left), we will use supervised classification model. We choice logistics regression function as our statistical function, and choice Backpropagation neural network (BP neural network) as training and prediction model.

The artificial neural network is a new type of information processing system, which is based on a preliminary understanding of the structure of the human brain and active systems. For a non-linear mapping capacity of an artificial neural network and a good soft topology [4]. The use of the BP algorithm of the multilayer neural network model is called the BP neural network. BP artificial neural network is the most used neural network, the most studied networks [5].

Another technique borrowed from machine learning in statistics is logistic regression. It is a classic algorithm for solving classification problems. This article will use a combination of logistic regression and neural network models to predict vehicle cornering behavior.

3 Introduction of data

In this paper, NGSIM data is selected as training data and test data. Researchers at the Next Generation Simulator (NGSIM) collected detailed data on vehicle trajectories from 101 U.S. South and Lankershim Avenue in Los Angeles, California, eastbound I-80 to Emeryville, California and Peachtree Street in Atlanta, Georgia. Data is collected through a network of synchronized digital cameras. GNVIDEO is a custom software application developed for the NGSIM program that records vehicle trajectory data from video. This vehicle trajectory data can provide the precise position of each vehicle in the study area every tenth of a second, so as to obtain the detailed position and lane position relative to other vehicles. [6].

In the NGISM data, we have obtained about 1.1 million pieces of raw data, and each piece of data contains 25 characteristics including steering behavior, vehicle position, and car-following conditions. First, we need to filter the raw data. By deleting the data that does not contain steering information or contain abnormal information, we obtained about 250,000 valid data. By convention, we choose 70% of the data as the training set to train the model; use 30% of the data as the test set to test the trained model.

4 Create the model

Before the model is established, the first task is to determine the variables. After analyzing 25 features in the existing data, after deleting the irrelevant information such as text information and number, we finally confirmed 15 features as input variables, as the Table 1 show

Table 1 Features chosen and their description

Feature Name	Description
Local X	Lateral (X) coordinate of the front center of the vehicle in feet with respect to the left-most edge of the section in the direction of travel.
Local Y	Longitudinal (Y) coordinate of the front center of the vehicle in feet with respect to the entry edge of the section in the direction of travel.
Global X	X Coordinate of the front center of the vehicle in feet based on CA State Plane III in NAD83. Attribute Domain Val
Global Y	Y Coordinate of the front center of the vehicle in feet based on CA State Plane III in NAD83.

v_length	Length of vehicle in feet
v_Width	Width of vehicle in feet
v_Class	Vehicle type: 1 - motorcycle, 2 - auto, 3 - truck
v_Vel	Instantaneous velocity of vehicle in feet/second.
v_Acc	Instantaneous acceleration of vehicle in feet/second square.
O_Zone	Origin zones of the vehicles, i.e., the place where the vehicles enter the tracking system. There are 11 origins in the study area, numbered from 101 through 111. Please refer to the data analysis report for more detailed information.
D_Zone	Destination zones of the vehicles, i.e., the place where the vehicles exit the tracking system. There are 10 destinations in the study area, numbered from 201 through 211. Origin 102 is a one-way off-ramp; hence there is no associated destination number 202. Please refer to the data analysis report for more detailed information.
Int_ID	Intersection in which the vehicle is traveling. Intersections are numbered from 1 to 4, with intersection 1 at the southernmost, and intersection 4 at the northernmost section of the study area. Value of "0" means that the vehicle was not in the immediate vicinity of an intersection and that the vehicle instead identifies with a section of Lankershim Boulevard (Section_ID, below). Please refer to the data analysis report for more detailed information.
Direction	Moving direction of the vehicle. 1 - east-bound (EB), 2 - north-bound (NB), 3 - west-bound (WB), 4 - south-bound (SB).
Preceding	Vehicle ID of the lead vehicle in the same lane. A value of '0' represents no preceding vehicle - occurs at the end of the study section and off-ramp due to the fact that only complete trajectories were recorded by this data collection effort (vehicles already in the section at the start of the study period were not recorded).
Space_Headway	Space Headway in feet. Spacing provides the distance between the frontcenter of a vehicle to the front-center of the preceding vehicle.
Time_Headway	Time Headway in seconds. Time Headway provides the time to travel from the front-center of a vehicle (at the speed of the vehicle) to the front-center of the preceding vehicle. A headway value of 99
Lane_ID	Current lane position of vehicle. Lane 1 is farthest left lane; lane 5 is farthest right lane. Lane 6 is the auxiliary lane between Ventura Boulevard on-ramp and the Cahuenga Boulevard off-ramp. Lane 7 is the on-ramp at Ventura Boulevard, and Lane 8 is the off-ramp at Cahuenga Boulevard.

According to our initial prediction, the output variable is the steering behavior of the vehicle. There are three possibilities, so our output variables are 3 categories.

First, we tried to simulate the data using a logistic regression model. We used the regularized linear regression cost function. The training result of the model is ideal, and the accuracy rate in the training set reaches 93.01%, while the accuracy rate in the test set reaches 93.02%. Since our sample size is already large enough, it can be proved that the logistic regression model can effectively predict vehicle steering behavior. However, because the logistic regression model is only suitable for linear problems, we used the Backpropagation Neural network model for the second operation.

This paper chooses one hidden layer network structure. The number of nodes in the hidden layer is usually determined based on experience. A common empirical formula is:

$$l = \sqrt{m+n} \quad (1)$$

Where l, n, and m are the number of nodes in the hidden layer, input layer, and output layer.

In this paper, m = 15, n = 3, which can be obtained from the formula (1), l = 4. So, the topology of the bp neural network established in this paper is 15-4-3, as the figure1 shows.

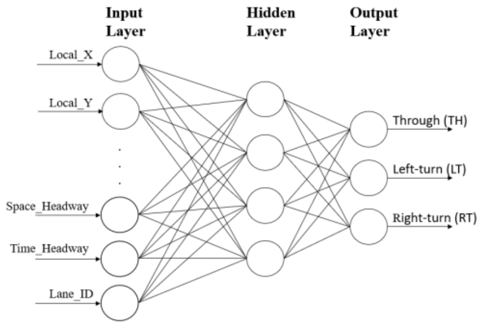

Fig.1 Schematic diagram of BP model

After training and testing, the correct rate of the BP Neural network model has also reached 93%. So, we can say that in the existing data set, there is a linear relationship between our related data and vehicle steering behavior.

5 Conclusion

The models and programming in this article are all completed by my pc. In the case of the same data set, the running time of the two models is quite different. BP Neural network takes nearly 5 hours due to the large amount of calculations, while the logistic regression model only takes 15 minutes. In future applications, we should choose the most suitable model. In the training, the correct rate of predicting the steering behavior of the vehicle reached 92.01%, and in the test, the correct rate reached 92.02%. This result shows that the model established in this paper can predict the steering behavior of vehicles more accurately.

6 Reference

[1] Haoyang Fan and Zhongpu Xia and Changchun Liu and Yaqin Chen and Qi Kong，An Auto-tuning Framework for Autonomous Vehicles[J].,arXiv,2018;1808.049

[2]宗长富,杨肖,王畅,张广才.汽车转向时驾驶员驾驶意图辨识与行为预测[J].吉林大学学报(工学版),2009,39(S1):27-32.

[3] Larasati, A., DeYong, C., & Slevitch, L. (2012). The Application of Neural Network and Logistics Regression Models on Predicting Customer Satisfaction in a Student-Operated Restaurant. Procedia - Social and Behavioral Sciences, 65, 94–99.

[4] Jianjun Wang1 , BaojieYan2 , kuanmin Chen3 . Some problems on expressway post-evaluation [J].Chang'an University's learned journal.2005.6

[5] Li, M., & Chen, W. (2012). Application of BP Neural Network Algorithm in Sustainable Development of Highway Construction Projects. Physics Procedia, 25, 1212–1217.

[6] https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories

7 Appendix-code

```
input_layer_size  = size(X_train,2);

hidden_layer_size = 4;   % 8 hidden units

num_labels = 3;          % 3 movements


m = size(X_train, 1);

y = Y_train(:, 1);


Theta1=rand(hidden_layer_size,(input_layer_size+1));

Theta2=rand(num_labels,hidden_layer_size+1);

nn_params = [Theta1(:) ; Theta2(:)];


% ==========Initializing Pameters =======

initial_Theta1 = randInitializeWeights(input_layer_size, hidden_layer_size);

initial_Theta2 = randInitializeWeights(hidden_layer_size, num_labels);


%=============train===============
```

```
options = optimset('MaxIter', 50);

lambda = 3;

initial_nn_params = [initial_Theta1(:) ;
initial_Theta2(:)];

costFunction = @(p) nnCostFunction(p, ...
                                   input_layer_size, ...
                                   hidden_layer_size, ...
                                   num_labels, X_train, y,
lambda);

[nn_params, cost] = fmincg(costFunction,
initial_nn_params, options);

Theta1 = reshape(nn_params(1:hidden_layer_size *
(input_layer_size + 1)), ...
                 hidden_layer_size, (input_layer_size +
1));


Theta2 = reshape(nn_params((1 +
(hidden_layer_size * (input_layer_size +
1))):end), ...
                 num_labels, (hidden_layer_size + 1));



%=================Predict============

pred = predict(Theta1, Theta2, X_test);

fprintf('\nTraining Set Accuracy: %f\n',
mean(double(pred == Y_test)) * 100);


function [J grad] = nnCostFunction(nn_params, ...
                                   input_layer_size, ...
                                   hidden_layer_size, ...
                                   num_labels, ...
                                   X, y, lambda)
        % for our 2 layer neural network
        Theta1 =
        reshape(nn_params(1:hidden_layer_size *
        (input_layer_size + 1)), ...
                 hidden_layer_size,
        (input_layer_size + 1));


        Theta2 = reshape(nn_params((1 +
        (hidden_layer_size * (input_layer_size +
        1))):end), ...
                 num_labels, (hidden_layer_size
+ 1));


% Setup some useful variables

m = size(X, 1);


J = 0;

Theta1_grad = zeros(size(Theta1));

Theta2_grad = zeros(size(Theta2));


a1 = [ones(m,1) X];

z2 = Theta1 * a1';

a2 = sigmoid(z2);

a2 = [ones(1,m); a2];

z3 = Theta2 * a2;

a3 = sigmoid(z3);


y_vect = zeros(num_labels, m);

for i = 1:m

        y_vect(y(i),i) = 1;

end


for i=1:m

        J = J + sum(-
1*y_vect(:,i).*log(a3(:,i))-(1-
y_vect(:,i)).*log(1-a3(:,i)));

end

J = J/m;

J = J +
lambda*(sum(sum(Theta1(:,2:end).^2))+s
um(sum(Theta2(:,2:end).^2)))/2/m;

Delta1 = zeros(size(Theta1));

Delta2 = zeros(size(Theta2));

for i=1:m

        delta3 = a3(:,i) - y_vect(:,i);

        VV = Theta2'*delta3;

        delta2 =
VV(2:end,:).*sigmoidGradient(z2(:,i));

        Delta2 = Delta2+ delta3*a2(:,i)';
```

```
        Delta1 =Delta1+ delta2 * a1(i,:);
    end
    Theta2_grad = Delta2/m;
    Theta1_grad = Delta1/m;
    Theta2_grad(:,2:end) =
    Theta2_grad(:,2:end) + lambda *
    Theta2(:,2:end) / m;
    Theta1_grad(:,2:end) =
    Theta1_grad(:,2:end) + lambda *
    Theta1(:,2:end) / m;
    grad = [Theta1_grad(:) ; Theta2_grad(:)];
end

function g = sigmoid(z)
    g = 1.0 ./ (1.0 + exp(-z));
end

function g = sigmoid(z)
    g = 1.0 ./ (1.0 + exp(-z));
end

function numgrad = computeNumericalGradient(J,
theta)

    numgrad = zeros(size(theta));
    perturb = zeros(size(theta));
    e = 1e-4;
    for p = 1:numel(theta)
        perturb(p) = e;
        loss1 = J(theta - perturb);
        loss2 = J(theta + perturb);
        numgrad(p) = (loss2 - loss1) /
    (2*e);
        perturb(p) = 0;
    end

end

function p = predict(Theta1, Theta2, X)
    m = size(X, 1);
    num_labels = size(Theta2, 1);

    p = zeros(size(X, 1), 1);

    h1 = sigmoid([ones(m, 1) X] * Theta1');
    h2 = sigmoid([ones(m, 1) h1] * Theta2');
    [dummy, p] = max(h2, [], 2);
end
```