Online Adaptation of Parameters using GRU-based Neural Network with BO for Accurate Driving Model

Zhanhong Yang
Zhanhong.yan@ibm.com
IBM Research
Tokyo, Japan

Satoshi Masuda
smasuda@jp.ibm.com
IBM Research
Tokyo, Japan

Michiaki Tatsubori
mich@jp.ibm.com
IBM Research
Tokyo, Japan

ABSTRACT
Testing self-driving cars in different areas requires surrounding cars with accordingly different driving styles such as aggressive or conservative styles. Calibrating a driving model (DM) makes the simulated driving behavior closer to human-driving behavior, and enable the simulation of human-driving cars. Conventional DM-calibrating methods do not take into account that the parameters in a DM vary while driving. These “fixed” calibrating methods cannot reflect an actual interactive driving scenario. In this paper, we propose a DM-calibration method for measuring human driving styles to reproduce real car-following behavior more accurately. The method includes 1) an objective entropy weight method for measuring and clustering human driving styles, and 2) online adaption of DM parameters based on deep learning by combining Bayesian optimization and a gated recurrent unit neural network. We conducted experiments to evaluate the proposed method, and the results indicate that it can be easily used to measure human driver styles. The experiments also showed that we can calibrate a corresponding DM in a virtual testing environment with up to 26% more accuracy than with fixed calibration methods.

CCS CONCEPTS
• Theory of computation → Bayesian analysis.

KEYWORDS
Online Adaptation, Bayesian optimization, Vehicle driving model

ACM Reference Format:

1 INTRODUCTION
Examples of human driving styles are aggressive, normal, and conservative. These styles vary among countries and areas. It is necessary to create a virtual environment with drivers from different countries and areas to fully test a self-driving car. For example, drivers in Japan have a relatively low accident rate than those in the USA [1]. Therefore, testing a self-driving car in the USA requires a more aggressive style than in Japan. The following two problems need to be solved to create such a testing environment:

1. A driver’s driving style is an abstract concept, it is necessary to develop a method of measuring such styles numerically so that all driver styles in one area can be represented to create a corresponding testing environment.
2. A driving model (DM) should simulate real human driving behavior.

A car-following model describes the movements of a following vehicle (FV) in response to the actions of the leading vehicle (LV) [10] and is used in a simulation environment such as Simulation of Urban Mobility (SUMO) [2]. In a DM, a driver’s behavior is controlled by a series of math equations. Setting the parameters in a DM to make the simulated behavior closer to actual human driver behavior is called calibrating a DM. A well-calibrated DM enables the simulation of an actual car, but the parameters of the DM are fixed in the simulation [6].

We propose a DM-calibration method for measuring real driving styles and for online adaptation of DM parameters using a gated recurrent unit (GRU)-based neural network (NN) to reproduce real car-following behavior more accurately. Figure 1 shows overview of the proposed method. The proposed method first uses the objective entropy weight method for measuring driving data [9] and clustering them. It then splits the driving data by using a short time window (e.g., 0.5 sec) and uses Bayesian optimization (BO) to search for optimized DM parameters for the time windows. It finally trains a GRU-based NN with the optimized DM parameters. We evaluated the proposed method by applying it to the
Krauss car-following [7] and Wiedemann car-following [8] models with the public dataset of driving data Next Generation Simulation (NGSIM), Interstate 80 freeway, collected in California, USA [3]. The results indicate that the proposed method is effective in a real simulation environment, i.e., SUMO. The results also indicate that our method can easily numerically differentiate human driver styles throughout a dataset. Therefore, our method could reproduce more accurate DM than conventional DM-calibration methods that do not take into account that the parameters in a DM vary while driving. DM consists of car-following model and lane-changing model in microscopic traffic simulation tool [5]. In this paper, we focus on car-following model as a DM.

In response to the problems and limitations discussed above, our contributions are as follow:

(1) Our DM-calibration method includes the objective entropy weight method for measuring human driving styles.
(2) Our method also involves adaptive calibration.
(3) The experimental results indicate that our method can reproduce more accurate DM in simulating a driver’s velocity trajectory than conventional DM-calibration methods.

2 PROPOSED METHOD

2.1 Measuring driving-style with objective entropy weight

Our proposed method includes the objective entropy weight method [9]. This method is based on the information provided by various attributes to find weights of the information entropy. The steps of calculating the weights are as follows:

(1) For one driving trajectory dataset, construct evaluation matrix E. Assuming the set includes trajectory data of m vehicles with ID = 1,2,…,m, a vehicle with ID = i, E = (e_{ij})_{m \times n}, where n = 6, is

\[
E = \begin{pmatrix} Vel_{mean,1} & Vel_{mean,2} & \cdots & Vel_{mean,m} \\ Vel_{var,1} & Vel_{var,2} & \cdots & Vel_{var,m} \\ Acc_{mean,1} & Acc_{mean,2} & \cdots & Acc_{mean,m} \\ Acc_{var,1} & Acc_{var,2} & \cdots & Acc_{var,m} \\ H_s,1 & H_s,2 & \cdots & H_s,m \\ H_f,1 & H_f,2 & \cdots & H_f,m \end{pmatrix}
\]

where Vel_{mean} is mean velocity, Vel_{var} is variance of velocity, Acc_{mean} is mean acceleration, Acc_{var} is variance of acceleration, H_s is average speed headway, and H_f is average time headway.

(2) Normalize E into (0,1) and obtain \( E_{nor} = (e'_{ij})_{m \times n} \). For column j, normalize E in accordance with the following equations:

\[
e_{ij} = \begin{cases} \frac{\max(e'_{ij}) - e_{ij}}{\max(e'_{ij}) - \min(e'_{ij})} & (e'_{ij} < 0), \\ \frac{e_{ij} - \min(e'_{ij})}{\max(e'_{ij}) - \min(e'_{ij})} & (e'_{ij} > 0). \end{cases}
\]

\[
\max(e'_{ij}) = \max(e'_{ij}, e'_{ij}, \ldots, e'_{mj})
\]

\[
\min(e'_{ij}) = \max(e'_{ij}, e'_{ij}, \ldots, e'_{mj})
\]

(3) For column j, calculate the percentage of data weight \( p_{ij} \):

\[
p_{ij} = \frac{e_{ij}}{\sum_{i=1}^{m} e_{ij}}
\]

(4) Calculate the entropy score \( ent_j \):

\[
ent_j = -\frac{1}{\ln(m)} \sum_{i=1}^{m} p_{ij} \ln(p_{ij})
\]

(5) Calculate the entropy weight vector \( W \):

\[
W = w_j = -\frac{1 - ent_j}{n} \sum_{i=1}^{n} ent_j
\]

For most cases, we consider that one driver drives the same vehicle in the dataset. Then, the exact driving style of vehicle ID=1 (also driver ID=1) in \( E_{nor} \) is measured by a score that is represented as \( s_i \):

\[
s_i = [e_{i1}, \ldots, e_{ij}, \ldots, e_{i6}] \cdot W
\]

The objective entropy weight method is used to divide and select the driving trajectory dataset.

2.2 Online adaptation of DM parameters

With the proposed method, different driving styles are in one dataset. To create an exact virtual testing environment, however, we need a method for reproducing a driver’s behavior. In simulation, a driver’s behavior is generally controlled using a DM. In this study, we used the Krauss and Wiedemann models as DM models. A previous study proposed many fixed DM-calibration methods [10]. However, the main limitations with these methods are that they do not take into account that the inner parameters can vary when driving (hereafter, these methods are called “fixed calibration methods”). The proposed method uses BO and a GRU-based NN. Figure 2 shows the architecture of the GRU-based NN. Assuming that our reproducing target is a subject vehicle with a leading vehicle for a car-following DM. The output should be a policy \( \pi (v_p, v_s) \) used for the NN, which inputs both the leading vehicle state \( v_p \) and subject vehicle state \( v_s \), e.g., subject vehicle’s speed trajectory in the past few steps. The proposed method predicts the next possible parameters, which minimizes the difference between the actual and simulated trajectories.

Algorithm 1 shows the steps of the online adaptation of DM parameters. Assuming the target is to reproduce the driving behavior of the subject vehicle by using a DM with \( M \) inner parameters sets \( pM(p_1, p_2, \ldots, pM) \) to be calibrated. The data used for reproduction can be one driver (assuming he/she has an ID=1, \( s_j \)). For driver i in the reproduction dataset, we assume the data from the subject vehicle as \( d_{s,i} \), and the data from the pre-coding as \( d_{p,i} \).

3 EXPERIMENTS

3.1 Dataset

We conducted experiments on the NGSIM, Interstate 80 freeway (I-80) dataset, and collected data between 4:00 p.m. and 4:15 p.m. on April 13, 2005. The study area was approximately 500 meters (1,640 feet) in length. The dataset consists of comma-separated values and the columns include vehicle ID, position (x, y), velocity, acceleration, following vehicle ID, and so on. The data represent
Dense Layer 1, 180 units
GRU Layer 2, 20 units
GRU Layer 1, 20 units
Figure 2: Gated recurrent unit (GRU)

Algorithm 1 Algorithm for online adaption of DM parameters for vehicle $i$
Input: $d_{s,i}, d_{p,i}$
Output: $\pi (v_p, v_s)$
1: repeat
2: $t=0, a, 2a...$
3: Split the $d_{s,i}, d_{p,i}$ into short time windows every $a$ steps, each window length is denoted as $L$, obtain a sliced dataset from $d_{s,i}, d_{p,i}$ as $d_p^{k=0,L}, d_p^{k=a,L+a}...; d_s^{k=0,L}, d_s^{k=a,L+a}...$, where $k$ is the time frame.
4: repeat
5: Loss = RMSE ($d_s^{k=j,L+j} = j, L+j + ms$)
6: For time step $j$, search for the best inner-parameter set $P_{M,j}$ by BO using $d_p^{k=j,L+j}, d_s^{k=j,L+j}$
7: until all $d_p^{k=j,L+j}, d_s^{k=j,L+j}$ have been found
8: until $d_{s,n}, d_{p,n}$ has been found
9: Train $\pi (v_p, v_s)$ with input $d_p^{k=j,L+j}, d_s^{k=j,L+j}$ and labels: $P_{M,j}$

Output Layer 1, $n$ units
(2 units for Krauss, 10 units for Wiedemann)
Dense Layer 1, 180 units
Gated recurrent unit (GRU)

moving vehicles every 0.1 second (one frame). To obtain sufficient trajectory data for the experiments, we selected 94 pairs of leading and following vehicles ($I$-80 Selected Data) that have over 70 seconds (700 frames) for their driving, from a total of 1475 vehicles in the dataset.

3.2 Driving-style score
We calculated driving-style scores $s_i$ for the I-80 Selected Data. Figure 3 shows the driving styles in the distribution. We divided the styles into three clusters on the basis of their percentile; cluster-0 (conservative), which is the bottom 25% (24 vehicles, driving score $<0.380$), cluster-1 (normal), which is the middle 50% (46 vehicles, $0.380 \leq$ driving score $\leq 0.564$) and cluster-2 (aggressive), which is the top 25% (24 vehicles, driving score $>0.564$).

3.3 Online adaptation

3.3.1 Vehicle data for Krauss model. We first applied the proposed method to each pair of leading and following vehicles in the I-80 Selected Data. The two car-following models simulated the following vehicle in accordance with the leading vehicle, so we applied the proposed method to the following vehicle. The trajectory data were split into 0.5-second (5 frames) short time windows, and the parameters for the sliced dataset were searched for by BO. We used the first 80% of the trajectory data for training the GRU-based NN with 0.1 for validation split as a hyper-parameter of training the NN. The remaining 20% of data were selected as the test sets for the GRU-based NN. For example, a pair of leading ID=1 and following ID=11 vehicles has records of 83.9-seconds (839 frames) for its trajectory in the dataset, the data were split into 167 (=839/5) time windows, and the NN training used 134 (=167×80%) and 33 (=167×20%) time windows. We used epochs $= 500$ and batch size $= 1$ as hyper-parameters of training, as a result of grid search of the hyper-parameters.

We evaluated the proposed method by comparing it with fixed calibration methods; one using the Krauss model with the initial default parameters, e.g., $r_t = 1.5$ and $t_l = 0.15$, (hereafter, the method is called default parameters) and with fixed parameters searched for by BO using 80% of the trajectory data (hereafter, this method is called fixed parameters). Note that this BO is only used in experiments and different from the BO in the proposed method. Figure 5 shows the results of default parameters with the velocity of the car-following simulation ($Vel_{Sim}$) and real data ($Vel_{Real}$) for the following vehicle ID=11 trajectory-data frames, Figure 6 shows the velocity on the same vehicle ID for fixed parameters, and Figure 7 shows the results of the proposed method, i.e., online adaptation of parameters. Table 1 shows the root mean square error (RMSE) between $Vel_{Sim}$ and $Vel_{Real}$. The proposed method differed from fixed parameters. We focused on the improvement of the proposed method from fixed parameters to evaluate the accuracy of the proposed method. In the following vehicle ID=11 case, RMSE of the fixed parameters was 1.033 and the one of proposed method was 0.666, therefore, RMSE improvement of the proposed method from the fixed parameters was 35.5% (=(1.033-0.666)/1.033) (hereafter, this improvement is called RMSE improvement for evaluation of the proposed method).

3.3.2 Applying Krauss and Wiedemann models to each vehicle pair. We next evaluated the proposed method by applying it to each vehicle pair (94 pairs of leading and following vehicle data) for both the Krauss and Wiedemann models. The data of each vehicle pair were used to train and test, the same as in the previous experiment for the Krauss model.
improvement between both car-following models were less than zero, parameters depended on the vehicle pair. The average ing low velocity.

gressive driving, which may be due to conservative driving involv-
one adaptation of conservative driving is more difficult than ag-
for cluster-2 (aggressive driving) was greater than those for cluster-0 (conservative driving). This indicates that for cluster-0 (conservative driving). This indicates that having a greater amount of training data than each vehicle pair. For example, the number of training data for vehicle ID=11 was only 134; however, cluster-0 had about 2500 time windows (=19 vehicles × 134 time windows).

5 CONCLUSION AND FUTURE WORK

The proposed method divided vehicle trajectory data in accordance with their driving styles and achieved more accurate than other DM-calibration methods. We applied the proposed method to two car-following models; however, we believe the method can also be applied to humanoid robots that imitate human behavior since such robots use models to simulate human actions.

REFERENCES