Instance-based Learning of Vehicular Performance Models

Ty Nguyen[1] and Tsz-Chiu Au[1]

Abstract— **Motion planning with predictable timing and velocity will enable a number of interesting applications such as autonomous intersection management. These planning algorithms depends on an accurate model of the performance of the vehicular controllers, which can be highly non-linear. Au et al. proposed a motion planning algorithm that is based on a performance model of the vehicle controllers. However, they assumed that the performance models are given for every road, and did not discuss how to build these models. In this paper, we propose an instance-based learning method to learn a performance model automatically, and compare it with the artificial neural networks. We argue that instance-based learning is suitable for this task because performance models for different roads are quite similar. An exploration strategy based on the principle of least effort is given to speed up the learning process. Our experiments showed that the instance-based learning method with distance-based exploration strategy has a faster rate than the artificial neural network methods.**

I. INTRODUCTION

Motion planning algorithms often assume an accurate model of a physical system for the reliable execution of the motion plans they generate. However, this assumption does not hold in many real world situations, causing errors in plan execution. For example, we previously considered the problem of controlling a mobile robot (e.g., an autonomous vehicle) to arrive at a given position on a one dimensional trajectory at a specific arrival time and a specific arrival velocity [1]. This motion control with *arrival requirements* is fundamental in a number of multi-robot systems, in particular autonomous intersection management (AIM) which coordinates vehicles to enter an intersection in unison, leading to a much lower traffic delay than traffic signals and stop signs [2]. In some sport games such as robot soccer, the question of whether a player can move to a target position at certain time and at certain velocity to hit a ball is important in role assignment and formation positioning [3]. The motion planning algorithm in [1] depends on an accurate model of the vehicle's performance in order to plan ahead to meet the arrival requirements. However, this work assumed the performance model is given, and did not discuss how to build the model. Since a vehicle will run in a variety of road conditions, we could not obtain the performance model for running on every possible road. Therefore, this is necessary to employ some machine learning techniques to conduct vehicle performance profiling automatically.

In this paper, we consider two approaches for learning a performance model of a vehicle. The first approach is artificial neural networks, which is a popular method for function approximation. The second approach is the instance-based learning approach, which relies on the fact that the performance models for different roads are quite similar. In addition, to improve the learning rate of these algorithms, we propose a new exploration strategy that gathers samples with the least effort first. We conduct experiments to evaluate these approaches and find out which approach has a faster learning rate. A fast algorithm for learning a performance model can reduce the model error that may have an impact on the performance of the motion planner which utilizes the performance model.

This paper is organized as follows. After presenting the related work in Section II, we give the definition of a performance model in Section III. Then we precisely define a longitudinal motion planning problem that utilizes a performance model in planning in Section IV. Section V gives the details of the two learning approaches, and Section VI describes the exploration strategy. Finally, we present experimental results and conclude this paper in Section VII and Section VIII, respectively.

II. RELATED WORK

Effectively modeling the acceleration behavior of vehicles plays a central role In a wide variety of transportation engineering applications. The acceleration profiles of a vehicles can be used to design roadways, to model vehicle behavior in crash simulation, or to estimate fuel consumption and emission. Vehicle accelerations are also important in planning problems, as discussed in the introduction. However, researchers have been struggling to develop models to predict the acceleration profiles of a vehicle. In practice, there are several difficulties in obtaining a set of precise equations describing the accelerating behavior of a vehicle under a particular traffic condition, including the imperfect vehicle dynamics, noise and the complexity in modeling the environment characteristics. Therefore, data-driven methods have been serving as an alternative approach to acquire a model of the accelerating behavior, and Machine learning is the key to build these models.

Data-driven methods can be divided into two main categories: parametric and nonparametric approaches. In parametric approaches, the model can fall into either kinematics model or dynamics model categories. To begin with, kinematics models consider the mathematical relationship between acceleration, speed, and distance that the vehicle has traveled. The most basic of kinematics models are the constant acceleration model, linear decay model [4] and dual-regime model [5], to name a few. These models basically

[1]School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, UNIST-gil 50, Eonyang-eup, Ulij-gun, Ulsan, Republic of Korea {tynguyen,chiu}@unist.ac.kr

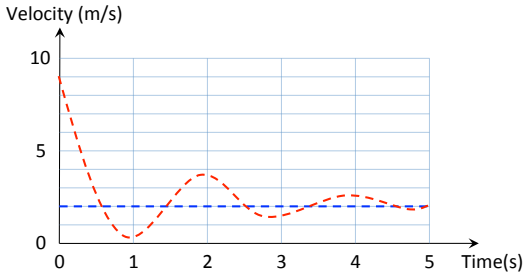
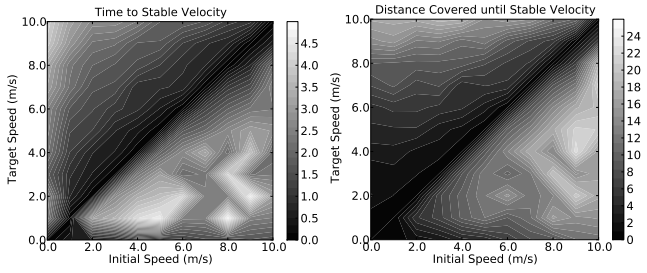


Fig. 1. An example of the velocity response of a PID controller. Overshooting occurs when the vehicle decelerates to a velocity that is close to zero.



(a) Stable Time (b) Stable Distance

Fig. 2. Stable time and stable distance for a particular vehicle under a particular environment. The light color means longer time and distance, as indicated in the bars beside the graphs.

attempt to empirically construct mathematical expressions that describe how the vehicle accelerates. However, the determinant factor of the acceleration process—the tractive force provided by the engine and the opposing resistance forces are ignored. For this reason, these kinematics models cannot provide reasonable fitting to field data.

On the contrary, both of the tractive effort and resistance forces which act on the vehicle's body and control the vehicle's motion are taken into account to develop vehicle dynamics models. Rakha et al. was the first to bring forth a constant power model and a variable power model to determine the performance of trucks [6]. Many efforts have been followed by [7] and [5] to calibrate the dynamics models. Although these dynamics models provide a good fit to the field data, it is hard to decide which breaking points are appropriate for different regimes, not to mention that these breaking points are subject to variation as data sets change. Besides, these models need intensive calibration before using them.

The downsides of the parametric models are twofold: First, the majority of parametric models only predict the maximum acceleration capabilities of a vehicle. Second, due to the limit of number of parameters, it is hard to estimate highly nonlinear terms and measurement noise. For these reasons, nonparametric estimation can be an appropriate alternative. Indeed, there are a few works on nonparametric estimation. For example, in [8], Kim and Oh adopted an Artificial Neural Network model to predict the next state of the vehicle given the current vehicle state, the current input steering angle of the wheels and the vehicle's velocity. The neural network model is associated with the hybrid learning scheme. In addition, Park et al. introduced a speed prediction algorithm, namely Neural Network Traffic Modeling-Speed Prediction, which is trained with the historical traffic data and capable of predicting the vehicle speed profile by using current traffic information [9].

III. PERFORMANCE MODELS

The goal of modeling vehicle performance is to enable long-term planning of vehicle's movement *without* knowing the details of vehicle dynamics and controls. This separation of high-level planning issues from the concerns of lower-level vehicle controls enables our planning procedures, called *setpoint schedulers*, to work with a wide variety of vehicle hardwares with different underlying control mechanisms. Longitudinal control of autonomous vehicles are usually achieved by throttle and braking systems coupled with sensors such as odometers and speedometers using PID-controllers. A setpoint is the target velocity given to the PID-controllers so as to control to vehicle to reach the setpoint. However, due to the complexity of the system, it is often hard to tune the PID gains to achieve a smooth transition after changing the setpoint. For example, if the autonomous vehicle at UT Austin decelerates from 9 m/s to 2 m/s, it will take 4.7 s to stabilize and the stable distance is 19.3 m—-a long stable time and stable distance when compared with acceleration. This problem is due to overshooting, as illustrated in Fig. 1, which is an intrinsic characteristics of vehicle dynamics.

Fortunately, for planning purpose it is *not* necessary to take every detail of the vehicular behavior into account. Given the current velocity v and a setpoint $\hat{v}$, the setpoint scheduler only needs to know how long the PID controllers will take to stabilize the velocity of the vehicle at $\hat{v}$ after setting the setpoint to $\hat{v}$, and how much the vehicle will move before its velocity is stabilized. Thus our approach relies on the estimation of two functions T^{stable} and D^{stable}, where $T^{\text{stable}}(v,\hat{v})$ is the time the vehicle takes to stabilize at $\hat{v}$ and $D^{\text{stable}}(v,\hat{v})$ is the distance the vehicle travels after setting the setpoint to $\hat{v}$ for a period of $T^{\text{stable}}(v,\hat{v})$. We call $T^{\text{stable}}(v,\hat{v})$ and $D^{\text{stable}}(v,\hat{v})$ the *stable time* and the *stable distance*, respectively. The *performance model* of the vehicle is the pair $(T^{\text{stable}}, D^{\text{stable}})$. Fig. 2 shows that the performance model in a table format.

The performance model is built via an empirical performance profiling of the PID controllers for the brake and throttle actuators of a vehicle. Our previous work assumed this profiling is given [10]. This paper discusses in detail how this profiling can be achieved by machine learning techniques.

IV. SETPOINT SCHEDULING PROBLEMS

Let us define a setpoint scheduling problem that utilizes a performance model. Suppose a vehicle is moving on a

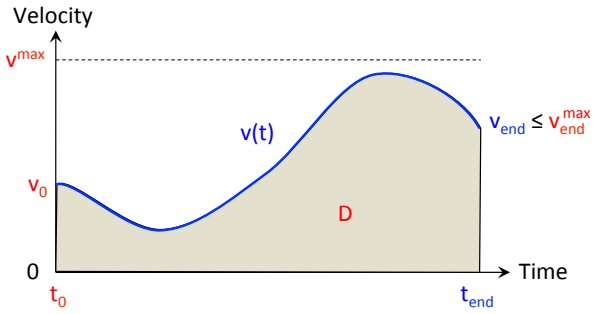


Fig. 3. The time-velocity diagram.

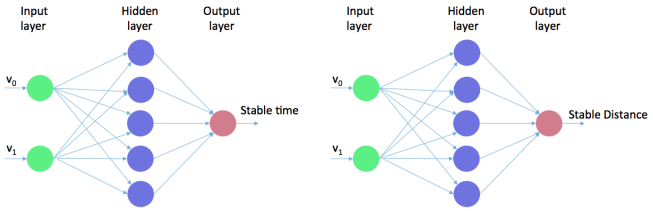


Fig. 4. A performance model represented by two artificial neural networks

one-dimensional trajectory such as a road. We are interested to control the vehicle to arrive at a destination on a road at a given arrival time t_{end} and at a given arrival velocity v_{end}. We define 1) the *initial configuration* as (t_0, v_0), 2) the *arrival configuration* as $(t_{\mathsf{end}}, v_{\mathsf{end}})$, and 3) the *road configuration* as $(D, a_{\mathsf{max}}, a_{\mathsf{min}}, v_{\mathsf{max}})$, where D is the road's length, v_{max} is the speed limit of the road, and a_{max} and a_{min} are the maximum and minimum acceleration, respectively. A *longitudinal motion planning problem* $\mathscr{P}_{\mathsf{valid}}$ is a 3-tuple $\langle (t_0, v_0), (t_{\mathsf{end}}, v_{\mathsf{end}}), (D, a_{\mathsf{max}}, a_{\mathsf{min}}, v_{\mathsf{max}}) \rangle$, where $t_0, v_0, t_{\mathsf{end}}$, and v_{end} are initial time, initial velocity, arrival time, and arrival velocity, respectively, such that $t_0 = 0$, $0 \le v_0 \le v_{\mathsf{max}}$, $0 < t_{\mathsf{end}}$, $0 \le v_{\mathsf{end}} \le v_{\mathsf{max}}$, $0 < D$, $a_{\mathsf{max}} \ge 0$, $a_{\mathsf{min}} \ge 0$, and $0 < v_{\mathsf{max}}$. Our planning task is to generate a *setpoint schedule* such that if the vehicle follows the schedule exactly, it will reach the destination while satisfying all requirements and constraints. We denote a *setpoint schedule* by $\tau(\cdot)$. If $\tau(\cdot)$ is a step function, $\tau(\cdot)$ can be represented by a list of pairs $\langle (t_0, \hat{v}_0), (t_1, \hat{v}_1), \ldots (t_n, \hat{v}_n) \rangle$, such that $\tau(t) = \hat{v}_i$ for (1) $t_i \le t < t_{i+1}$ for $0 \le i < n$ and (2) $t_i \le t$ for $i = n$. In this paper, we assume all setpoint schedules are step functions.

One way to visualize what we are trying to achieve is to take a look at the time-velocity diagram in Figure 3. In this diagram, the line is a *velocity function* which denotes the velocity of the vehicle over time. A velocity function $v(\cdot)$ is *constructible* if there exists a setpoint schedule $\tau(\cdot)$ such that if the vehicle follows $\tau(\cdot)$, the velocity of the vehicle is $v(\cdot)$. A velocity function $v(\cdot)$ is *feasible* if it satisfies the following constraints:

1) $v(t_0) = v_0$;
2) $0 \le v(t) \le v_{\mathsf{max}}$ for $t_0 \le t \le t_{end}$ (i.e., the velocity cannot exceed the speed limit of the road or be negative at any point in time);
3) $\int_{t_0}^{t_{end}} v(t)\, dt = D$, where t_{end} is the arrival time (i.e., the distance traveled must be D); and
4) $v(\cdot)$ is constructible.

A setpoint schedule $\tau(\cdot)$ is *feasible* if the velocity function constructed by $\tau(\cdot)$ is feasible.

The objective of a longitudinal motion planning problem $\mathscr{P}_{\mathsf{valid}}$ is to check whether a feasible setpoint schedule $\tau(\cdot)$ exists, and generate $\tau(\cdot)$ if it exists. Alternatively, $\mathscr{P}_{\mathsf{valid}}$ is called an instance of the *validation problem*, in which we want to *validate* the given arrival configuration $(t_{\mathsf{end}}, v_{\mathsf{end}})$ by checking whether $(t_{\mathsf{end}}, v_{\mathsf{end}})$ is reachable by a feasible setpoint schedule.

V. INSTANCE-BASED LEARNING ALGORITHMS

According to our problem formulation, a performance model has two functions: T^{stable} and D^{stable}. Our learning task is to estimate these functions by function approximators. One popular function approximator is artificial neural network (ANN). It has been shown that ANN with hidden layers can be used to approximate any functions. One common and well-known algorithm for training ANN with hidden layers is the backpropagation algorithm. Fig. 4 shows the ANNs that we used as the performance model. We use two ANNs, one for T^{stable} and the other for D^{stable}, and they work independently. The input nodes are v_0 and v_1, which are the starting velocity and the setpoint, respectively. Both ANNs have a hidden layer with 5 nodes. Given the error in the output nodes, we will use the backpropagation algorithm to adjust the weights on the edges in the ANNs.

The weights of the connections between neurons are usually initialized with random values before training. However, this fails to utilize the prior knowledge in the domain. In our learning task, there is a strong similarity between the performance models on different roads, since a performance model is largely dominated by the vehicle's controller rather than the road conditions. Therefore, we propose to consider an instance-based learning approach that utilizes the performance model on another road while learning the performance model. This approach involves two steps: First, obtain a performance model of a *reference road*, which we consider typical among the set of roads we consider. Second, modify the performance model according to the "samples" the vehicle collected by interacting with the road. Here, a sample is the stable time t_s and the stable distance d_s after setting a new setpoint. Given a sample (t_s, d_s), we modify the stable time function and the stable distance function, which are internally represented by two tables, according to the following update rule. Suppose the previous velocity is v and the new setpoint is v_{new}. Let t_s and d_s be the stable time and the stable distance we *measured* after setting v_{new} and until the velocity settles at v_{new}. Here we assume the measurement is *exact*, meaning that there is no error in the measurement. Let t_s' and d_s' be the stable time and the stable distance in the performance model before the application of the update rule. Then the update rule is that for every pair

(v_1, v_2) of velocities in the tables.

$$T^{\mathsf{stable}}(v_1,v_2) = \begin{cases} t_s & \text{if } v_1 = v \text{ and } v_2 = v_{\mathsf{new}} \\ T^{\mathsf{stable}}(v_1,v_2) + \alpha A \Delta_t & \text{otherwise} \end{cases}$$

and

$$D^{\mathsf{stable}}(v_1,v_2) = \begin{cases} d_s & \text{if } v_1 = v \text{ and } v_2 = v_{\mathsf{new}} \\ D^{\mathsf{stable}}(v_1,v_2) + \alpha A \Delta_d & \text{otherwise,} \end{cases}$$

where α is the learning rate, $\Delta_t = t_s - t_s'$, $\Delta_d = d_s - d_s'$, and $A = (v_2 - v_1)^2$. The term A gives the entries closer to the top-left corner and the bottom-right corner of the tables a larger weight. It is because the values closer to these corners are usually bigger due to the fact that there is large difference between the current velocity and the setpoint, and the vehicle will take a more time and distance to stabilize at the setpoint.

In our experiments, we pre-train the ANNs in the same environment that yields the reference model, so that the initial weights of the edges are not random. Hence the starting ANNs are an accurate performance model that is roughly equal to the reference model used by the instance-based learning.

VI. MIN-DISTANCE EXPLORATION STRATEGY

Given the assumption that all measurements are exact, we can easily show that in our instance-based learning approach, the performance model will converge to the true model when the vehicle measures the stable times and the stable distances of *all* possible ways the velocity can change. However, the order of samples the vehicle collects depend on its *exploration strategy*. In general, since the performance model $(T^{\mathsf{stable}}, D^{\mathsf{stable}})$ will be used repeatedly for many different motion planning episodes, each of them uses different parts of the model, we want the estimated model to be as *complete* as possible, meaning that we should spread out the samples so that no part of the model will be ignored forever. To facilitate the sampling process, the vehicle employs an exploration strategy, which determines the sequence of samples so as to minimize the time it takes to learn the performance model.

Here we consider exploration strategies that have the following three characteristics: First, the vehicle will sample at the discrete velocity values only. Let $\mathbb{V} = \{n \times d\}_{n=\{0..m\}}$ be the set of discrete velocity values where d is the discretization step and $d \times m$ is the maximum velocity, which is either the speed limit of the road or the top possible velocity of the vehicle. Second, after a vehicle deliberately changes its setpoint to measure the stable time and the stable distance, it will *immediately* start another measurement right away the vehicle is stabilized at the new setpoint. The result is that the ending velocity of a sampling step is always the starting velocity of the next sampling step. This way the vehicle can avoid the idle time between two samples. Third, the vehicle should avoid collecting the same sample again, because doing the same measurement twice is redundant since we assume that the measurement is exact.

Based on these characteristics, an exploration strategy can be considered as a sequence of setpoints $\langle v_0, v_1, v_2, \ldots, v_n \rangle$, where v_0 is the initial velocity of the vehicle, and v_i is the next setpoint after the vehicle stabilizes at v_{i-1}, for $i \geq 1$. The vehicle, starting with the initial velocity v_0, will first set its setpoint at v_1 and then measure the stable time and stable distance when its velocity stabilizes at v_1. After the measurement, it immediately sets its setpoint at v_2 for the second measurement. Then the process continues until the last setpoint v_n. To ensure completeness, this sequence of setpoints should be chosen to exhibit the property that the set of all possible consecutive pairs of setpoints (i.e., $\{(v_i, v_{i+1})\}_{i=0..(n-1)}$) is exactly the set of all possible pairs of *different* velocities in $\mathbb{V}$ (i.e., $\{(v,v') : v, v' \in \mathbb{V} \text{ and } v \neq v'\}$). Thus, under this exploration strategy, the performance model will converge to the true one.

A faster rate of convergence to the true performance model is important because the vehicle may not have enough time to acquire all the measurements before it uses the performance model for motion planning. It will be quite helpful to the motion planner if the vehicle can get fairly accurate performance model early on. We believe that the order of setpoints in an exploration strategy will greatly affect the speed of learning. In particular, we hypothesize that if an exploration strategy takes the measurements that require shorter distance of travel first, the rate of convergence to the true performance model will be faster. Therefore, we propose an exploration strategy as follows: when choosing the next setpoint for the next measurement after a vehicle stabilizes at v, always choose the one that has the minimum stable distance according to the reference model (i.e., choose $\arg\min_{v'}\{D^{\mathsf{stable}}(v,v') : D^{\mathsf{stable}}(v,v')$ has not been measured yet.$\}$) If there is no such setpoint, choose the next setpoint randomly. We call this strategy the *min-distance* exploration strategy. While this greedy approach makes sense, we still need to conduct experiments to validate our hypothesis in order to see how well this approach works.

VII. EXPERIMENTS

We conducted experiments to evaluate the learning methods and the exploration strategy. The experiments are based on a simulation platform we developed using PyGame[1], a Python library that supports 2-D simulation. In the simulation, a vehicle is running along a straight road with a slope. We have to set various parameters including the angle of the slope, the air resistance, the friction force coefficients, etc. in the simulation. Fig. 5 shows a screenshot with a very short slope. In our experiments, the slope is much longer—long enough to finish the learning episode. For each parameter setting, our goal is to learn a performance model when the vehicle runs on the slope.

We implemented our ANNs using ByBrain [11], a machine learning library for Python. The ANNs in this experiment are two feedforword neural networks that use the sigmoid

[1] `http://www.pygame.org`

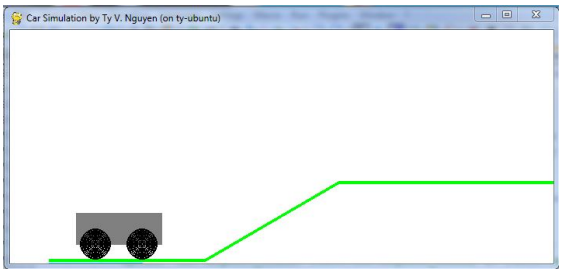


Fig. 5. A screenshot of the simulator with a short slope.

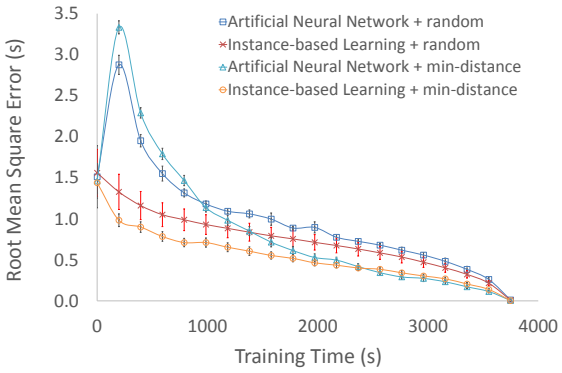


Fig. 6. RMSEs of stable time versus training time

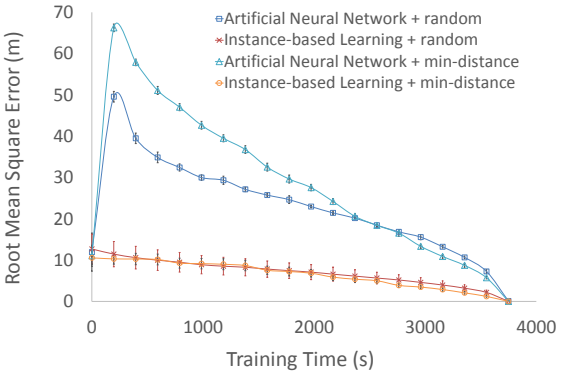


Fig. 7. RMSEs of stable distance versus training time

activation function, as shown in Fig. 4. Both ANNs consist of two input nodes, which correspond to the starting velocity and the target velocity. We examined the effects of the number of nodes at hidden layer and the number of hidden layers, by allowing the number of nodes at hidden layer to vary between 2 and 10 and the number of hidden layers to vary between 1 and 10. After a thorough evaluation, we settled with one hidden layer and five nodes. As discussed in Section V, the ANNs are pre-trained in the same environment as the one generated the reference model. The ANNs are updated by using the backpropagation function in PyBrain.

The implementation of the instance-based learning is based on the update rule as described in Section V. The learning process starts with a performance model of the reference road, and updates the performance model according to the update rule. We saw that larger the learning rate can result in a faster but more uncertain convergence. In this study, we chose the learning rate α equal to 0.5.

In our experiments, we randomly chose 30 sets of parameters of the reference roads as well as the target roads we intend to learn. A set of parameters includes θ, C_r, ρ which are the slope of the road, the coefficient of rolling friction and the density of air, respectively, where $0 \leq \theta \leq 60$, $0.001 < C_r \leq 0.303$ and $1.146 \leq \rho 1.423$. For each set of parameters, we learnt an accurate performance model of the reference roads by exhaustively testing all possible pairs of starting velocities and setpoints. Then we used it as the reference model of the instance-based learning approach. We also ran the ANNs on the reference road to initialize the weights of the ANNs.

After that we started the training phrase. We considered two exploration strategies, one is the min-distance exploration strategy in Section VI and the other is a random strategy that randomly chooses the next setpoint in the learning process. We applied both strategies to both ANNs and the instance-based learning. The vehicle used the exploration strategies to try out different setpoints, collect the stable times and stable distances, and update the performance models. Eventually, all possible pairs of starting velocities and setpoints have been tried and the exploration process stops.

After the exploration process stopped, we obtained the final performance model. This final performance model is solely based on the measurement, and therefore it is the true performance model. Then we can use this final performance model to compute the model errors during the learning process. This allows us to see how quickly the learning approaches can reduce the model error. This model error can be quantified by the root mean square errors (RMSE) of the difference of two matrices. Given an intermediate performance model that is acquired during learning, we can subtract the final performance model from an intermediate performance model to get an error table. For the entries in the intermediate performance model that are based on actual measurement, we will set the corresponding entry in the error table to zero. Then we computed the root-mean-square of all entries in the error table.

We plotted the RMSEs as the performance models evolved during the learning process over time. The result is shown in Fig. 6 and 7. Notice that each data point in the figures is an average of the 30 RMSEs of the 30 trials, and the error bars represent the 95% confident intervals. As can be seen, the instance-based approach outperformed the ANN approach in terms of the learning rates in both the stable time and the stable distance. Moreover, the min-distance exploration strategy does have some positive influence on the learning rate. The only exception is that when the ANN approach was used with the min-distance exploration strategy, the learning rate is comparable to the best strategy when the training time is large. But overall the instance-based learning approach, together with the min-distance exploration strategy, had the best performance.

VIII. CONCLUSIONS

For precise vehicle control, motion planning algorithms often rely on a performance model that accurately describes how the vehicle interacts with the road. In this paper, we focused on learning a behavior-based performance model of a vehicle with non-linear control. Instance-based learning approach is suitable for this task because the behavior of a vehicle is quite similar on different roads. We presented a novel instance-based learning approach to learn a performance model of a vehicle's controller. The approach directly adapts the performance model of the reference road for a different road, using an update rule. In essence, our instance-based learning approach is quite similar to transfer learning, through the source domain and the target domain are different in terms of the road settings only. We compared this approach to artificial neural networks that are pre-trained on the same reference road. An exploration strategy based on the principle of least effort was proposed to speed up the learning process. According to our experiments, both instance-based learning approach and artificial neural network can eventually learn the true performance model given enough samples. However, the instance-based learning approach has a faster learning rate than neural networks. In the future, we intend to remove the assumption that all measurements are exact and investigate how to select the best performance model as the reference model. We will also conduct experiments to study how the model error affects the performance of motion planning.

REFERENCES

[1] T.-C. Au, M. Quinlan, and P. Stone, "Setpoint scheduling for autonomous vehicle controllers," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 2055–2060.

[2] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research (JAIR)*, March 2008.

[3] P. MacAlpine, E. Price, and P. Stone, "Scram: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2015.

[4] D. R. Drew, "Traffic flow theory and control," Tech. Rep., 1968. [Online]. Available: http://trid.trb.org/view.aspx?id=115219

[5] G. H. Bham and R. F. Benekohal, "Development, evaluation, and comparison of acceleration models," in *81st Annual Meeting of the Transportation Research Board, Washington, DC*, 2002.

[6] H. Rakha, I. Lucic, S. H. Demarchi, J. R. Setti, and M. V. Aerde, "Vehicle dynamics model for predicting maximum truck acceleration levels," *Journal of transportation engineering*, vol. 127, no. 5, pp. 418–425, 2001. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-947X(2001)127:5(418)

[7] J. Searle, "Equations for Speed, Time and Distance for Vehicles Under Maximum Acceleration," SAE Technical Paper, Tech. Rep., 1999. [Online]. Available: http://papers.sae.org/1999-01-0078/

[8] Y. U. Yim and S.-Y. Oh, "Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction," *Vehicular Technology, IEEE Transactions on*, vol. 53, no. 4, pp. 1076–1084, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1317211

[9] J. Park, D. Li, Y. L. Murphey, J. Kristinsson, R. McGee, M. Kuang, and T. Phillips, "Real time vehicle speed prediction using a neural network traffic model," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2991–2996. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6033614

[10] T.-C. Au and P. Stone, "Motion planning algorithms for autonomous intersection management," in *AAAI 2010 Workshop on Bridging The Gap Between Task And Motion Planning (BTAMP)*, 2010.

[11] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "Pybrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.