

Inverse Reinforcement Learning via Neural Network in Driver Behavior Modeling

QiJie Zou, Haoyu Li

BeiDou High Precision Positioning Service Technology
Engineering Laboratory
Information Engineering College, Dalian University
DaLian, China
Jessie_zou_zou@163.com, 15542541936@163.com

Rubo Zhang

Mechanical and Electrical Engineering College
Dalian Minzu University
DaLian, China
zhangrubo@dlmu.edu.cn

Abstract—Inverse Reinforcement Learning (IRL) is formulated within the framework of Markov decision process (MDP) where we are not explicitly given a reward function, but where instead we can observe an expert demonstrating the task that we want to learn to perform. Then the expert as trying to maximize a reward function that is expressible as a linear combination of known features specifying the reward function. However, in autonomous driving tasks, due to the difference of scene factor, such as obstacle and weather, the state spaces are frequently large and demonstrations can hardly visit all the states. It's hard to get an optimal policy with RL method to express driver behavior model based on the reward which recovered with IRL method in this large-scale state space. In this paper, we focus on driving behavior modeling with IRL method which introduces the convolutional neural network to extract the associated state feature automatically, and express the policy by neural network to generalize the expert's behaviors. Experimental results compared with the traditional end-to-end method on simulated vehicle show that the accuracy of decision-making greatly improved in the train curve, and in the new curve scene with a large number of unvisited state, this method shows a perfect generalization efficiency.

Keywords—Human Factors and Human Machine Interaction; Advanced Driver Assistance Systems; Automated Vehicles

I. INTRODUCTION

As a mobile robot that completed by many parts, such as environment perception, task planning, behavioral decision, behavior planning, and vehicle operation, etc. Autonomous vehicles remain unfulfilled, mainly for the technical reason include software algorithm, hardware, and infrastructure. In addition to technical reason, there is also the question of trust in autonomous system and driver preference, driver preference. Thus, current autonomous vehicles still require autonomous driving systems and human pilots to cooperate with each other to accomplish driving tasks. In such a cooperative driving vehicle where such an autonomous driving system and a human driver coexist, whether it is to better quantify the driver's information for decision making by an intelligent system or to provide personalized service to people by distinguishing drivers from each other. Driver modeling are all

essential steps. As a major research area in the field of autonomous driving, based on different emphasis, drivers are modeled in different degrees and aspects. Driver modeling refers to the research method that represents the physical attributes of the driver by simulating the behavior of the driver operating the car, in order to reproduce the driving process [1-2]. The indicators used include: steering wheel angle, speed and other parameters [3], and related algorithms include: control theory [4-8], Hidden Markov Model [9-10], machine learning and other methods. In this paper, we focus on the problem of driver behavior modeling in a typical scenario, when the system actively intervenes the driver in the cooperative driving system. In other words, it is to model human behavior with respect to the task of driving a vehicle in a typical scenario. In order to comprehensively identify abnormal driving behaviors without over-conservative and radical judgments, it is necessary to consider diversified attributes in all aspects.

Reinforcement learning as a kind of machine learning algorithm to describe the driving behavior model is more accurate. Reinforcement learning builds on Markov's decision-making process and takes action to capture feedback signals (called reward) of the nature of the assessment from the environment and maximize long-term returns by interacting with the environment. Reinforcement learning can improve the learning ability of autonomous systems. And reinforcement learning has a natural advantage for the sequential decision-making problem of continuous state like driving decision [11] and has been widely used in autonomous system to enable the learning ability of agent [12-14]. However, in practical problems and scenarios, the reward function is a problem that is difficult to define and describe. Reward function settings often need to balance too many different needs, such as: car distance, curb, pedestrians, a reasonable speed and change frequency. The traditional method of setting the proportion of these factors is often dependent on the experience of researchers. Inverse Reinforcement Learning (IRL) could make a learning agent to discover the underlying rewards from a bunch of demonstrated examples of a desired behavior

Compared to behavior clone which is similar to supervised learning, it directly imitates the experts' trajectory. IRL first obtains the reward function that enables the experts to

Supported by the National Natural Science Foundation of China (Grant No. 61673084).

demonstrate the optimal policy, and then indirectly completes the acquisition of the expert policy by using the reinforcement learning algorithm. IRL is originally introduced in [15]. Abbeel and Ng then proposed a method called apprenticeship learning method, this method is based on weighting of the state features, obtaining a reward function by learning from an expert demonstration, and the optimal policy learned with this reward function is in the vicinity of the expert demonstration policy. On this basis, the maximum margin planning (MMP) algorithm [16] also adopts the same idea: using the linear state feature to represent the reward function and make the performance of a policy better than other policies by margin. And then establish an objective function based on demonstration and prediction and use the sub-gradient method to minimize it. With the help of inverse reinforcement learning, we could recover the unknown reward function which is weighted sum of the state features. However, this state feature is still a man-made basement, which is sometimes not well described by man. The convolutional neural network is then introduced here to extract the state feature components in the state.

Comparing with the classical machine learning method, a large amount of data is preprocessed to become a feature by means of human prior knowledge, and then it is used in the corresponding algorithm. The performance of those algorithm depends on the feature, then the generalization of the policy obtained by these algorithms is poor. The benefits of an end-to-end approach: by reducing manual preprocessing and subsequent processing, the model is imported from the original to the final output as much as possible, giving the model more room to fit the model based on how much data is automatically adjusted. In this paper, we use the IRL approach for modeling driver behavior in autonomous driving to describe the driver's behavior policy, the CNN is used to complete the feature extraction for the state-action pairs. In the algorithm, the neural network is used to generalize to the expert demonstration to the untracked area in the state space, and a non-linear policy representation is adopted for the driver's driving policy. As can be seen from the experimental results, the neural network is used to represent the policy. The algorithm have a very good generalization efficiency for the new driving curve scene with many non-demonstrative states.

The article is structured as follows: Section 2 describes the problem and relevant subproblem, introduces the state feature extraction with CNN method, the inverse reinforcement learning which includes the calculation of reward function and corresponding policy. In section 3, the results of experiment verify presented method. Section 6 draws the final conclusion.

II. PROBLEM FORMULATION

In autonomous driving tasks, due to the numerous scene factors which influence the driving, such as obstacle and weather, it's hard to get an optimal policy with RL method to express driver behavior model in this large-scale state space. To be precise, it is very difficult and time consuming to consider all possible corner cases to describe the state feature which is used to constitute the reward, and it's also difficult to obtain a definitive policy based on the reward. In the following

section, we present an approach to automatically extract the state features about the driving decision and learn the reward function based on those features for large datasets of demonstration trajectories in the context of autonomous driving, based on the restored reward, we use a perfect policy representation to make the learning progress more effective and intuitive.

A. State Feature Extraction.

In this article, a driving simulator from Udacity [17] is used to collect driver data and realize autonomous driving of virtual vehicles. Simulator is divided into two modes of data collection and autonomous driving, the former by the driver to operate a virtual vehicle to generate driving data, the data include dashboard camera images and control data (steering angle, throttle, brake, speed) is shown in Table 1. The steering angle which we focus on is a variable that floats between -1 and 1, and as expert data to obtain model with the IRL method. The latter modes allow the virtual vehicle to accept control data generated from the model to accomplish autonomous driving. Then Compare the differences between autonomous driving based on the driver model and actual driving, we can obtain the driver behavior model efficiency.

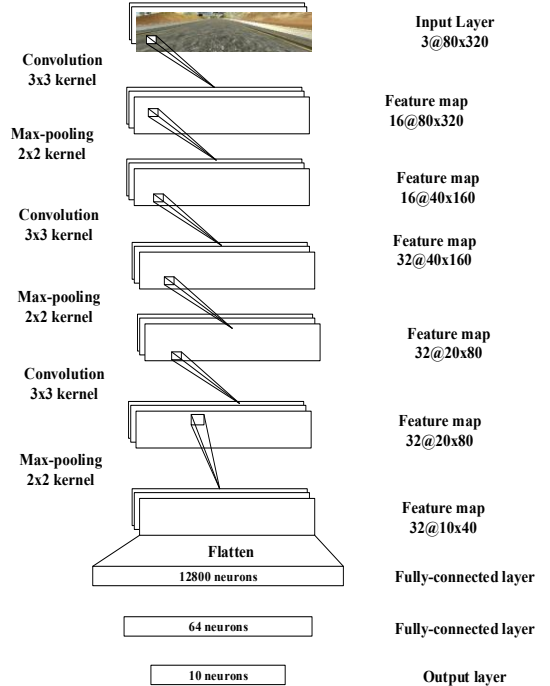


Fig. 1. Decision-related Feature Extraction Network

Driving the scene due to the weather, the scene, the different objects, the driving scene has many possibilities, so in order to correctly express the driving state, we use the state feature to describe the state space. The relevant driving state feature extraction, we use the convolutional neural network to extract the characteristics of the driving environment of the picture, the network structure is as Fig 1. The input of the network is camera images, the output is a vector of 1×10 , then,

the state feature at time t is described as $\phi(s_t) = \{\phi_i\}_{1 \leq i \leq k}$. In the state feature extraction network training process, the network need to add a layer of single neurons fully connected layer as a new output layer after the original output layer, then the steering angle in the control data is used as label data for the training process.

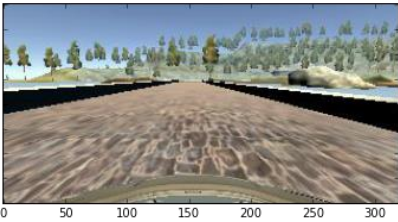
With training optimization of the new network structure, a network capable of outputting the correct steering angle is obtained, where the convolutional network can extract those state features relating to steering angle decisions in environmental conditions. The action space does not take a big action space with continuous values but introduce a researcher's transcendental experience through the

establishment of a fuzzy controller to convert the original continuous turning angle into several fuzzy intervals. The action space is $A = \{u_1, u_2, u_3, u_4, u_5\}$, respectively: Do not turn to the straight line, turn left slightly, turn right slightly, turn left vigorously, turn right vigorously.

B. Model-free maximum margin planning

Before obtain the optimal policy to describe the driver model, we need to deal with these MDP decision problems by using inverse reinforcement learning to find the unknown reward function. Inverse reinforcement learning can recover the unknown reward function for which optimal policy matches the demonstrations.

TABLE I. EXAMPLE DRIVING DATA

Camera Images	Control Data			
	Steering Angle	Throttle	Brake	Speed (km/h)
	-0.923744	0.668526	0.0	29.98325

The agent observes N_T trajectories which consist of L_i state-action pairs in each trajectory:

$$D_E = \{(s_1, a_1), (s_2, a_2), \dots, (s_M, a_M)\} \quad (1)$$

where $M = \sum_{i=1}^{N_T} L_i$. IRL obtain the reward function by linear combination of k state feature components f_k .

$$r(s, a) = \theta^T f(s, a) = \sum_{k=1}^K \theta_k f_k(s, a) \quad (2)$$

where $\forall (s, a) \in S \times A$. The feature components of the state indirectly describe the perception of the environment. The action-value function $Q^\pi(s, a)$ under the policy can be rewritten as follows:

$$Q^\pi(s, a) = \theta^T \mu^\pi(s, a)$$

$$\text{with } \mu^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \mid s_0 = s, a_0 = a \right] \quad (3)$$

μ_π is called as the feature expectations of the policy π , they completely determine the expected sum of discounted rewards for acting according to that policy [15]. The expectations of the features f_i based on the policy π are stated below: [16]

$$\mu_i^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t f_i(s_t, a_t) \mid s_0 = s, a_0 = a \right] \quad (4)$$

The maximum margin planning method [16] is to find a policy on which the margin between the characteristic expectation and the expert characteristic expectation is less than a certain value, as follow:

$$t = \|\mu^\pi - \mu_E\| < \delta \quad (5)$$

During the iteration, each state on the relevant trajectory is set to a different value according to the choice of action on different policies, called the loss function. With the loss function (generally set to 0 or greater), the total return of the expert policy is always greater than the return of the other learned policies.

$$\min_{\theta} \frac{1}{2} \|\theta\|_2^2 + C \sum_{i=1}^{N_T} \xi^{(i)} \quad (6)$$

$$s.t. Q_{\theta}^{\pi_E}(s_t^{(i)}, a_t^{(i)}) + \xi^{(i)} \geq \max_{\pi} Q^{\pi}(s_t^{(i)}, a_t^{(i)})$$

where $Q^{\pi}(s_t^{(i)}, a_t^{(i)})$ is the action value under a learned policy in state s . $Q_{\theta}^{\pi_E}(s_t^{(i)}, a_t^{(i)})$ is the action value of expert demonstration. Slack variable ξ is given for each expert trajectory in order to allow constraint violations for a penalty, and it is supposed to be small. By solving this quadratic programming problem, θ is further obtained and then used to construct a return function.

C. Policy Representation

A policy can be viewed as a connection between the state in the state space and its corresponding action. If we are dealing with a large-scale state space, it's hard to get a

definitive policy, and with a perfect policy representation, the learning progress could be much more effective and intuitive.

Universal approximation theorem of the neural network shows that the feedforward neural network can express the function from one finite dimensional space to another finite dimensional space with any precision as long as it has a linear output layer and a hidden unit with an activation property and has excellent generalization ability. It is therefore chosen to describe policies for tactical expression to optimize the learning process in IRL. After completing the description of the state space and the action space, we then use neural networks to describe the policy. the state feature at time t is described as $\phi(s_t) = \{\phi_i\}_{1 \leq i \leq k}$, and the state-action feature at time t is defined as $f(s_t, a_t)$, which is a matrix of $k \times n$, described as:

$$f(s_t, a_t) = [v_1, v_2 \dots v_n] \quad (7)$$

$$\text{with } v_i = \begin{cases} \phi(s_t)^T & \text{if } a_t = u_i \\ 0 & \text{otherwise} \end{cases}$$

where u_i is the action of action space, this function $f(s_t, a_t)$, which contains state information and action selection is used as a basis function for the reward function.

We build a three-layer neural network to fit the mappings $g: s \rightarrow R^{n \times 1}$, thus linking the state s_t to related action values $Q(s_t, u)$, and the mapping is described as:

$$g(s_t) = \begin{bmatrix} Q(s_t, u_1) \\ Q(s_t, u_2) \\ \vdots \\ Q(s_t, u_n) \end{bmatrix} \quad (8)$$

The network structure shown in Figure 2.

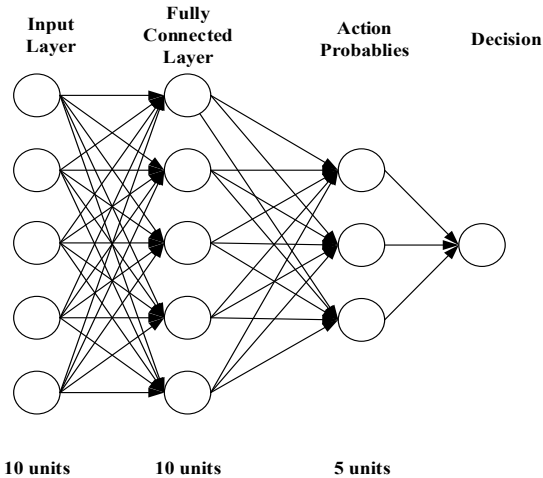


Fig. 2. The Nonlinear Policy Representation Neural Network

The activation function between hidden layer and output layer is sigmoid function $\text{sig}(x) = \frac{1}{1 + e^{-x}}$.

In summary, the mathematical expressions in neural networks are as follows:

$$z = w^{(1)}x = w^{(1)}[1, \phi_t]^T;$$

$$h = \text{sigmoid}(z); \quad (9)$$

$$g_w(s_t) = \text{sigmoid}(w^{(2)}[1, h]^T)$$

The input of the NN is $\phi_t = \phi(s_t) \in R^{k \times 1}$, the output is (8), and every $g(s_t, a_t)$ can be rewritten as $Q(s_t, u_t)$.

With the policy representation network, we select the action based on the action values obtained by different actions for a certain state s . Specific, the expression as shown below:

$$\pi_\Omega(s_t, a_t) = P(a_t | s_t; \Omega) = \frac{\exp(\eta g(s_t, a_t))}{\sum_{u \in A} \exp(\eta g(s_t, u))} \quad (10)$$

where η is the Berman parameter to control the randomness of action selection

D. Neural Network Policy Iteration

To obtain the optimal policy of the corresponding reward function, it is necessary to optimize the nonlinear neural network iteration. We use the SARSA algorithm to update the Q value of the current policy, the update rules are as follows:

$$Q^\pi(s_t, a_t) = Q^\pi + \alpha [r_\theta(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)] \quad (11)$$

where $Q(s, a)$ is the previous neural network output value. α is learning rate to express the speed of learning. γ is discount factor. This Q value is used as target Q to update and optimize the weight of the neural network by the means of back propagation algorithm. The NN error $J(\Omega)$ which represents the difference between the output of the given input and the target value is defined as follows:

$$J(\Omega) = -\frac{1}{N} \sum_{t=1}^N [\hat{Q}^\pi(s_t, a_t) \circ \log Q^\pi(s_t, a_t) + (1 - \hat{Q}^\pi(s_t, a_t)) \circ \log(1 - Q^\pi(s_t, a_t))] \quad (12)$$

$$+ \frac{\lambda_2}{2N} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\omega_{j,i}^{(l)})^2$$

where N is the number of all state-action pairs under current policy. L is the number of network layer. s_l is the number of units in layer l , the last one is the regularization terms to prevent over-fitting. With the gradient descent method is adopted to minimize the error function, so the complete algorithm is presented as follow:

The IRL via NN algorithm

Input: the expert demonstration data $D = \{(s_1, a_1) \dots (s_M, a_M)\}$

1. Extract the state feature with CNN and use the state feature $\phi(s_t)$ to express the state s_t .
2. Obtain the expert trajectory $D_E = \{(s_1, a_1) \dots (s_M, a_M)\}$ for $s_t = \phi(s_t)$
3. Compute the expert's feature expectations μ_E as in (4)
4. Randomly generate an initial neural policy π_1
5. Set $i=1$
6. **While** $i>0$ **do**
7. Execute the current policy π_i , and generate a sequence of state-action pairs ξ_i to compute the current μ_i
8. Obtain θ by (6), and compute t_i as (5)
9. **if** $t_i \leq 0$ **then**
10. Terminate
11. **end if**
12. Compute the reward and using the SARSA algorithm to update the Q-values of the sequence as in (11).
13. Minimize the cost function $J(\Omega)$, to obtain the optimal neural weight, which represent the current optimal policy π^{i*}
14. Set $i=i+1, \pi^i = \pi^{i*}$
15. **end while**

Output: The optimal policy π^i

III. EXPERIMENT AND RESULT ANALYSIS

In this paper, the demonstration is the driver's driving decision data in the driving curve scenario. Based on the driving demonstration data, the reward function is learned by means of the algorithm of the article, then the mapping from the environment state to the driving decision action is obtained.

The driving trajectory of steady driving driver is taken as demonstration data. To increase the trajectory of the demonstration data, we add the driving task from the destination to the origin. The distribution of demonstration data for every frames is shown in Fig.3.

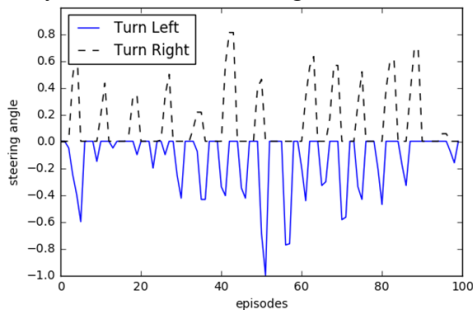


Fig. 3. The Steering Angle Distribution for Every Frames

In this paper, for curved driving scenes, there are two situations: left turn and right turn. And each iteration is updated with 200 state-action pairs as a episode to update the policy. In the process of algorithm updating, the reward function is updated by MMP method, the calculation method is to obtain the error between the feature expectation of the current policy and the feature expectation of the expert demonstration.

Mathematical expressions are shown as (5). The change of margin is shown in Fig 4.

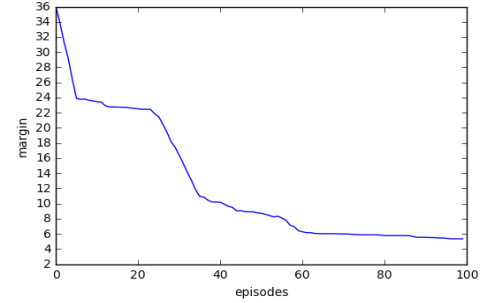


Fig. 4. The Changes of margin during the learning episodes

where the overall margin is declining, and eventually stabilize at 5. Hence, the margin between a learned policy was gradually approaching the expert policy, and this is exactly the learning objective. The nonlinear neural policy is trained for the state-action pairs collected under the current policy. The result to minimize the cost $J(\Omega)$ is shown in Fig 5.

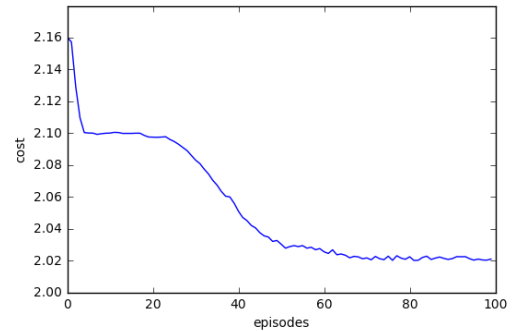


Fig. 5. The Changes of cost during the learning episodes

For the current driving task, the L2 error between IRL and demonstration is 10.401. and the error between E2E [18] and demonstration is 12.066.

For the driving task, the vehicle turns right from destination to origin, The L2 error between IRL and demonstration is 8.191. And the error between E2E and demonstration is 11.897.

For the new driving task, the vehicle driving from origin to destination, the decision distribution as shown in Fig 6. The L2 error between IRL and demonstration is 65.809. and the error between E2E and demonstration is 68.293.

IV. CONCLUSION

This paper presents a driver behavior modeling method, that the autonomous system could understand the expert's decision behavior in executing one specific task by means of learning the underlying reward functions from the expert demonstrations. This method is under the framework of inverse reinforcement learning. Firstly, we adopted convolutional neural network to extraction state feature, and optimized the algorithm with neural network to represent the driving decision policy, then, we adopted MMP method to learn the reward, Finally, we updated the neural network which represent the decision policy using the learned reward function.

The experiment results show the feasibility and the robustness of this method, the agent could make more accurate decisions than the traditional E2E method. The agent not can make parallel driving decision in demonstration trajectory state but complete the decision tasks with unvisited state. Future work will be concentrated on policy representation, the fuzzy neural network could introduce human prior knowledge, then adopt fuzzy neural network to represent the driving decision policy is our next step.

REFERENCES

- [1] XueJin Xu, Development Status and Direction of Driverless Vehicles [J]. Shanghai Auto, 2007, 7(40): 40-43.
- [2] Sun Zhen-Ping. Self - driving vehicle intelligent control system[D]. ChangSha: National University of Defense Technology, 2004
- [3] Murphy-Chutorian E, Trivedi M M. Head pose estimation and augmented reality tracking: An integrated system and evaluation for monitoring driver awareness[J]. IEEE Transactions on intelligent transportation systems, 2010, 11(2): 300-311.
- [4] Reddy R N, Ellis J R. Contribution to the simulation of driver-vehicle-road system[R]. SAE Technical Paper, 1981.
- [5] Guo K, Ding H, Zhang J, et al. Development of a longitudinal and lateral driver model for autonomous vehicle control[J]. International journal of vehicle design, 2004, 36(1): 50-65.
- [6] Peng H, Tomizuka M. Preview control for vehicle lateral guidance in highway automation[J]. TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF DYNAMIC SYSTEMS MEASUREMENT AND CONTROL, 1993, 115: 679-679.
- [7] Sharp R S, Peng H. Vehicle dynamics applications of optimal control theory[J]. Vehicle System Dynamics, 2011, 49(7): 1073-1111.
- [8] Na X, Cole D J. Game-theoretic modeling of the steering interaction between a human driver and a vehicle collision avoidance controller[J]. IEEE Transactions on Human-Machine Systems, 2015, 45(1): 25-38.
- [9] He L, Zong C, Wang C. Driving intention recognition and behaviour prediction based on a double-layer hidden Markov model[J]. Journal of Zhejiang University-Science C, 2012, 13(3): 208-217.
- [10] Minoura K, Watanabe T. Driving support by estimating vehicle behavior[C]//Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE, 2012: 1144-1147.
- [11] Momennejad I, Russek E M, Cheong J H, et al. The successor representation in human reinforcement learning[J]. bioRxiv, 2016: 083824.
- [12] Jaradat M A K, Al-Rousan M, Qadani L. Reinforcement based mobile robot navigation in dynamic environment[J]. Robotics and Computer-Integrated Manufacturing, 2011, 27(1): 135-149.
- [13] Miljković Z, Mitić M, Lazarević M, et al. Neural network reinforcement learning for visual control of robot manipulators[J]. Expert Systems with Applications, 2013, 40(5): 1721-1736.
- [14] Kober J, Bagnell J A, Peters J. Reinforcement learning in robotics: A survey[J]. The International Journal of Robotics Research, 2013, 32(11): 1238-1274.
- [15] A.Y. Ng, S.J. Russell, Algorithms for inverse reinforcement learning, in: International Conference on Machine Learning, ICML, 2000, pp. 663-670
- [16] Ratliff N D, Bagnell J A, Zinkevich M A. Maximum margin planning[C]//Proceedings of the 23rd international conference on Machine learning. ACM, 2006: 729-736.
- [17] <https://github.com/udacity/self-driving-car-sim>
- [18] Bojarski M, Del Testa D, Dworakowski D, et al. End to end learning for self-driving cars[J]. arXiv preprint arXiv:1604.07316, 2016.

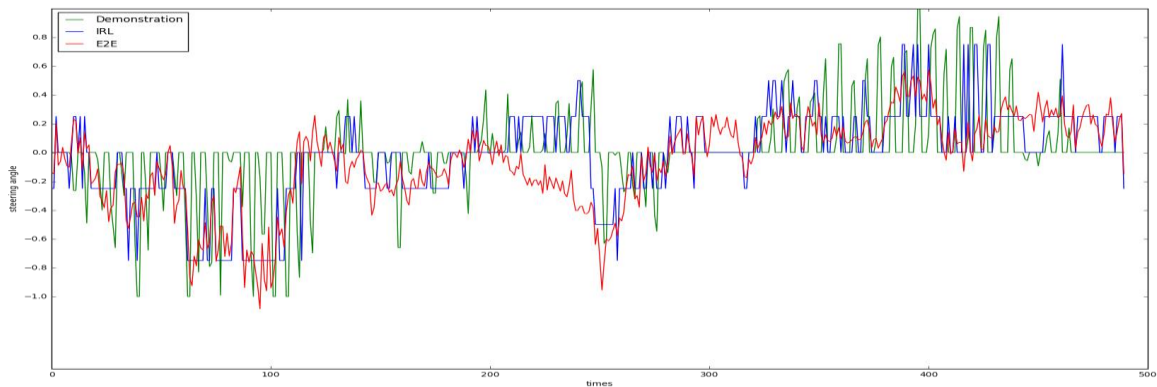


Fig. 6. The Decision Distribution of Various Methods in Test Scene