LEARNING THE REPRESENTATION OF BEHAVIOR STYLES WITH IMITATION LEARNING

Anonymous authors
Paper under double-blind review

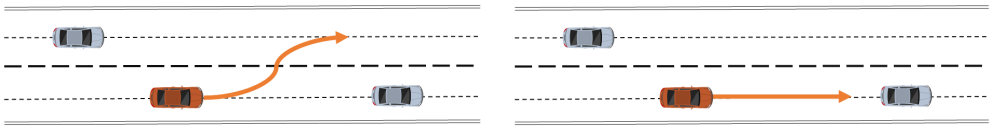
ABSTRACT

Imitation learning is one of the methods for reproducing expert demonstrations adaptively by learning a mapping between observations and actions. However, behavior styles such as motion trajectory and driving habit depend largely on the dataset of human maneuvers, and settle down to an average behavior style in most imitation learning algorithms. In this study, we propose a method named style behavior cloning (Style BC), which can not only infer the latent representation of behavior styles automatically, but also imitate different style policies from expert demonstrations. Our method is inspired by the word2vec algorithm and we construct a behavior-style to action mapping which is similar to the word-embedding to context mapping in word2vec. Empirical results on popular benchmark environments show that Style BC outperforms standard behavior cloning in prediction accuracy and expected reward significantly. Furthermore, compared with various baselines, our policy influenced by its assigned style embedding can better reproduce the expert behavior styles, especially in the complex environments or the number of the behavior styles is large.

1 INTRODUCTION

Deep reinforcement learning (DRL), which combines reinforcement learning and deep neural networks, has achieved rapid development in the last few years. Research institutions as well as industrial companies have applied this technology to many domains, such as games (Atari games and Go) (Silver et al., 2016; Van Hasselt et al., 2016), robotics control (Zhang et al., 2015) and autonomous driving (González et al., 2016). However, as the DRL algorithms optimize the models according to the reward function, it may not always generate desirable behaviors due to the inferior local convergence or inappropriate reward design, especially in the real world problems where the reward function is difficult to clearly define.

Imitation learning (IL) methods have the potential to cover this shortage, with which an agent learns to perform a task from expert demonstrations by mimicking a mapping between observations and actions instead of maximizing the accumulated reward (Osa et al., 2018). In these methods, the behavior style such as the motion trajectory and operation habit is determined depending on the dataset of expert demonstrations. In practise, the real world problems such as self-driving vehicles (Munoz et al., 2009), assistive robots (Ikemoto et al., 2012) and human-computer interaction (Ross & Bagnell, 2010) may need a large amount of demonstrations and these demonstrations are usually collected from multiple human experts who may have different behavior styles. For the conventional IL algorithms, the imitated behavior tends to converge into an average of the multiple behavior styles in the demonstrations when the imitation model is trained. However, demonstrations from different experts can show significant variability because different human experts typically employ different policies according to their own styles and habits, and learning an average behavior style maybe unacceptable. For instance, in a driving scenario with a leading vehicle, aggressive drivers may make the decision to change lanes. While some drivers are relatively conservative and will maintain a stable following distance with the front vehicle by controlling the velocity, as illustrated in Figure 1. In these situations, the average behavior reproduced by imitation learning may not express any kind of expert styles, or even worse results in an non-convergent policy (e.g., crashing the leading car in the scenario of Figure 1).



(a) (b)

Figure 1: Driving behaviors of human: (a) Lane-changing; (b) Car-following.

There are some algorithms considering behavior style in imitation learning. A classical way to learn from demonstration is Dynamic Bayesian Network (Eppner et al., 2009), which does not use deep learning. Another way is the constrained methods, such as Probabilistic Principle Component Analysis (Perico et al., 2019), which can take into account additional constraints such as speed limits and obstacles. Although these algorithms can reproduce the behavior style from a small number of demonstrations, due to a number of assumptions about the structure of sensory and control signals, such traditional methods are not scalable to more multi-dimensional and multi-modal systems, such as systems involving tactile sensors and raw images. Alternatively, methods based on Generative Adversarial Imitation Learning (GAIL) combine supervised and reinforcement learning (Ho & Ermon, 2016), which produces similar behaviors to the expert demonstrations in a simulation environment. Based on this, Information Maximizing GAIL (Info-GAIL) (Li et al., 2017), which can embed differences in behavior style in latent space using mutual information, has been used to produce different driver models in an autonomous driving environment. However, InfoGAIL involves sampling a latent code at the beginning of each trial. While when the simulated surrounding vehicles are initialized with the real driving environment information (such as velocity and position), the random sampling of latent codes may not ensure consistency between the policy's subsequent actions and the driver's behaviors. To address this limitation, a Burn-InfoGAIL algorithm (Kuefler & Kochenderfer, 2017), built on Info-GAIL, draws latent features directly from a learned, inference distribution. In addition, this burn-in policy must take over from the point at which a specific expert demonstration ends.

Building upon the above researches, this paper proposes a novel imitation learning framework named Style Behavior Clone (Style BC), which can learn the representation of various behavior styles from expert demonstrations. Not only can the model automatically learn a style embedding that is meaningful to represent the behavior styles, but also accurately reproduces different expert behaviors. Our approach is an extension of behavior cloning (BC) (Bojarski et al., 2016) and different from current style involved imitation learning algorithms which try to infer the style embeddings through inputting the behavior sequences, our method try to solve this problem in a more direct way: the correct embeddings should improve the final performance of BC and thus designing a framework which can adjust the style embeddings and the policy network simultaneously according the final training performance.

To be specific, this paper introduces a hypothesis that each policy has a latent feature c, i.e. style embedding, which can determine its behavior style. Therefore, in order to improve the prediction accuracy of the actions belonging to different styles, we need an accurate c to guide the behavior. Our method is inspired by the word2vec (Mikolov et al., 2013) algorithm which is used to embed the words in natural language processing (NLP). The word2vec algorithm try to compute the word embedding through optimizing the model to predict the contexts accurately, and correspondingly, Style BC tries to compute the behavior style embedding through optimize the model to predict the action accurately. Finally, it is worth noting that our algorithm updates style embedding by making the distinction of behaviors more clear, while Info-GAIL and related deformations disentangle the latent c by inferring from the sequences of state-actions pairs.

The main contributions of this paper are summarized as follows: (i) we propose a novel style behavior cloning approach that learns the various representation of behavior styles from expert demonstrations; (ii) the style embedding which can provide valuable guidance for the generation of style policies is obtained automatically; (iii) finally, we empirically demonstrate the effectiveness of our research on a Grid-World environment, cartpole control tasks from OpenAI gym, and a collection

of environment for decision-making in autonomous driving named highway-env. We show that our model can learn to represent different kinds of behavior styles by exploring the style embedding.

2 PROBLEM FORMULATION

In this section, we start with a brief introduction to the background including Markov Decision Process (MDP) and style Markov decision process (style MDP). Then we provide a basic objective function of behavior cloning to learn human behaviors from demonstrations. However, due to the expert demonstrations may collect a variety of behaviors, the simple supervised policy from offline demonstration may not have the ability to reproduce these motion styles.

2.1 BACKGROUND

RL problems are often modeled as Markov Decision Processes (MDPs) which are defined as a tuple $\mathcal{M} = (S, A, P, R, \gamma)$. S and A represent states and actions respectively. P is a state transition probability function, which can also be represented as $P(s'|s, a)$. Further $R : \ S \times A \rightarrow \mathbb{R}$ is a reward function measuring the performance of agents and γ is a discount factor for future rewards. Given an MDP $\mathcal{M}$, the goal of the agent is to maximize the expectation of the discounted accumulative reward $J_R = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\right]$ automatically by finding an optimal policy π^*.

In this paper, we assume that the behavior policy is a mixture of behavior styles $\prod_E = \{\pi_{B_1}(.|s, c_1), \pi_{B_2}(.|s, c_2) \ldots \pi_{B_n}(.|s, c_n)\}$, where c is a style embedding that can determine the behavior style of policy. The objective is to reproduce a policy $\pi(a|s, c)$ as an approximation of π_{B_i}, and we use the tuple (S, A, C, P, R, γ) to define this process, named style MDP.

2.2 BEHAVIORAL CLONING

BC as a representative imitation learning method seeks the best policy that can minimize the action prediction error in demonstrations. Traditional BC method learns a policy through supervised learning and the only requirements are pairs of input sensory observations associated with expert actions. The objective is to minimize the distribution discrepancy between the expert demonstrations and the policy, which can be formulated as follows:

$$\min_{\pi} \mathbb{D}\left[\rho_\pi(\cdot|s) \ || \ \rho_E(\cdot|s)\right], \tag{1}$$

where s represents the environment state and $\mathbb{D}$ is the discrepancy measure like KL divergence which is a common objective function in imitation learning. In addition, ρ_π and ρ_E depict the state action distribution generated by policy π and human demonstration E, respectively.

Human demonstrations can show significant variability because these demonstrations might be collected from users with different skills and habits. Since the policy is tied up to the demonstration in this objective function, in training the BC tends to converge into an average of various human demonstrations or a fixed behavior style when some behavior style dominates the overall demonstrations. However, sometimes we are more concerned with how to reproduce the variation of human behavior styles.

Therefore, inspired by style MDP, we try to improve the prediction accuracy of the model by learning the style embedding automatically, and leverage this encode to explore the variation of behavior styles.

3 PROPOSED METHOD

To learn the style embedding that captures the semantics of human behavior, our main insight is that the representation of styles can be reflected by their motions on the environment. In this paper, we assume that different style episodes in expert demonstrations are generated by a specific style policy $\pi_B(a|s, c)$. And what Style BC tries to do is to find the right behavior style embedding c for each episode in demonstration as well as learn a policy that can output the same action conditioning

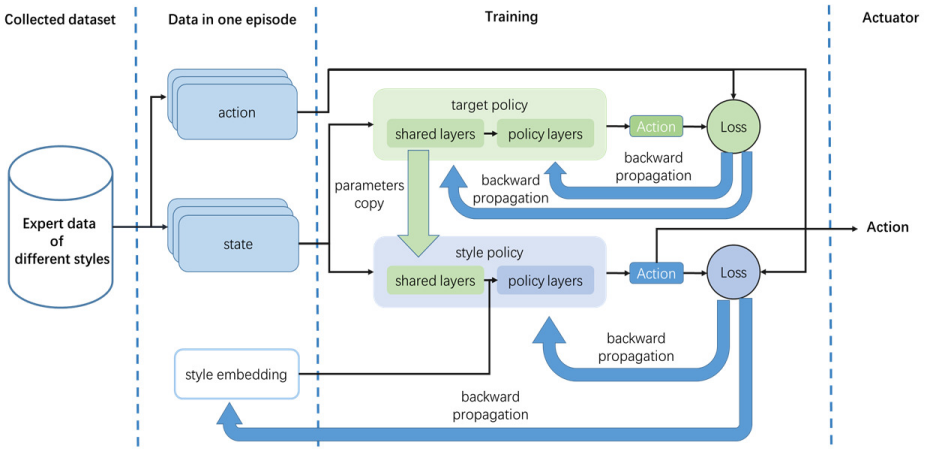


Figure 2: Overview of the framework of our proposed method.

on the that c and the state as $\pi_B(a|s,c)$ does. Figure 2 illustrates the overall framework of our method, which can be divided into two parts: learning the behavior style embedding and learning the imitation policy based on that embedding. The first part is implemented through the backward propagation process in the bottom of Figure 2. And for the learning of the imitation policy based on the embedding vector, we introduce an additional target policy to learn the representation of the state in the demonstration, which can also make the optimization process of the imitation policy more stable. An outline for the Style BC is shown in Algorithm 1..

3.1 The learning of the behavior style embedding

Firstly, in order to initialize the behavior style vectors C, we have to determine the number of the behavior styles N before training. We can determine it in two ways: 1) if the number of the episodes in the demonstrations is not large, we can set N to be the number of the episodes in the demonstrations; 2) otherwise if the number of the episodes in the demonstration is large, we can determine a proper value of N according to human domain knowledge, and we will introduce a method to select a right behavior embedding for each episode in the demonstrations Section 3.3. We take N as a hyper-parameter and this is the only domain knowledge related parameter in our framework.

After selecting the style vector c for each episode, we have to optimize this vector to help the model predicting the actions in this episodes more accurately. We leverage the widely used supervised

Algorithm 1: Style BC

Input: N: the number of the predetermined styles, E: the expert demonstration set, θ_{target} and ω_{target}: the network weights of the shared layers and the policy layers in target network, θ_{style} and ω_{style}: the network weights of the shared layers and the policy layers in style network

1 Initialize θ_{target}, ω_{target}, θ_{style} and ω_{style}, random initialize N behavior style vectors.
2 Select an expert episode e from E, and determine the style vector of this episode according to Eq. 6
3 Update the target network on e according to Eq. 4
4 Update the style vector c and the policy layers ω_{style} of style policy according to Eq. 2 and Eq. 3.
5 Update the shared layers θ_{style} of style policy according to Eq. 5

learning process for this purpose. As illustrated in Figure 2, the network named style policy is responsible for this process and it can be divided into two parts: the shared layers parameterized with θ_{style} and the policy layers parameterized with ω_{style} . We introduce the imitation learning loss to update c and the optimization can be formulated as:

$$\min_{c} \mathbb{D}\left[\pi_{(\theta_{style},\omega_{style})}(\cdot|s,c) \mid\mid \pi_B\left(\cdot|s,c\right)\right], \tag{2}$$

where $\pi_B\left(\cdot|s,c\right)$ is the action distribution calculated from the episode e with behavior style vector c.

3.2 The learning of the imitation policy

Next, we will introduce the process of optimizing the imitation policy, which is denoted as style policy in Figure 2. As mentioned above, the two network modules of it named shared layers and policy layers are parameterized with θ_{style}, and ω_{style} respectively. For θ_{style}, the optimization process is similar to the process in optimizing the behavior style embedding c, which can be formulated as:

$$\min_{\omega_{style}} \mathbb{D}\left[\pi_{(\theta_{style},\omega_{style})}(\cdot|s,c) \mid\mid \pi_B\left(\cdot|s,c\right)\right], \tag{3}$$

This optimization is the same as in the common behavior clone algorithm and it is intuitive to introduce it here as we want the style policy to mimic the expert policy. For θ_{style}, we introduce a target policy to update it, the target policy is a common behavior clone network with the same network construction as the style policy, except for that its input includes the state only and will not contain the style embedding. The shared layers and the policy layers of it are paremeterized with θ_{target} and ω_{target} respectively. In training, the parameters of the shared layers in style policy are copied from the ones in target policy. We use the target policy here because we take the outputs of the shared layers of the target network as a high level representation of the state and copy it to the style policy can make the optimization process of the style policy faster and more stable. The optimization process of θ_{style}, θ_{target} and ω_{target} can be formatted as:

$$\min_{(\theta_{target},\omega_{target})} \mathbb{D}\left[\pi_{(\theta_{target},\omega_{target})}(\cdot|s) \mid\mid \pi_B\left(\cdot|s\right)\right], \tag{4}$$

$$\theta_{style} = k * \theta_{style} + (1-k) * \theta_{target} \tag{5}$$

where k is the hyper-parameter used in the soft update of θ_{style}, and $\pi_{(\theta_{target},\omega_{target})}$ is the target policy network parameterized by θ_{target} and ω_{target}. Note that the input of the target policy network in Eq. 4 is only the state s, which is different from the input (including both the state s and the behavior style vector c) of the style policy network in Eq. 3

3.3 Selecting the behavior style vector for each episode

In this section, we will explain the method for selecting the behavior style vector when the number of the episodes is large and thus we have to predetermined the number of the behavior styles before training. After selecting an episode from the expert demonstrations, we will compute the prediction accuracy of the actions in this demonstrated episode with the style policy based on each behavior style vector c in C, and select the one with the highest prediction accuracy as the behavior style vector of this episode. To be specific, the behavior style vector c^e for episode e is determined through Eq. 6, where $P_{\pi_{(\theta_{style},\omega_{style})}}(e|c)$ is the prediction accuracy of style policy on the data in episode e taking the behavior style vector c as input.

$$c^e = \arg\max_{c \in \mathcal{C}} P_{\pi_{(\theta_{style},\omega_{style})}}(e|c) \tag{6}$$

4 Experiments

In this section, we aim at investigating (i) whether Style BC can reproduce the different behavior styles from expert demonstrations, and (ii) whether style embedding can guide and determine the generation of a specific style policy. To evaluate our approach, we conduct extensive experiments on several widely used environments: a navigation task in Grid-World, one classic control environment in OpenAI Gym (Kumar, 2020), and finally, a minimalist environment for decision-making in autonomous driving named highway-env (Leurent, 2018).

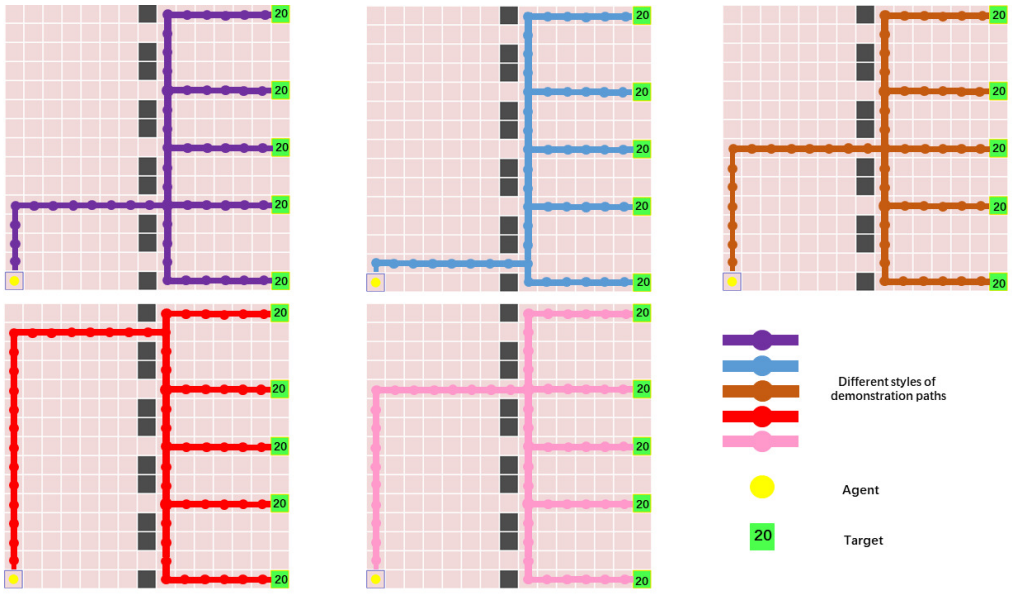
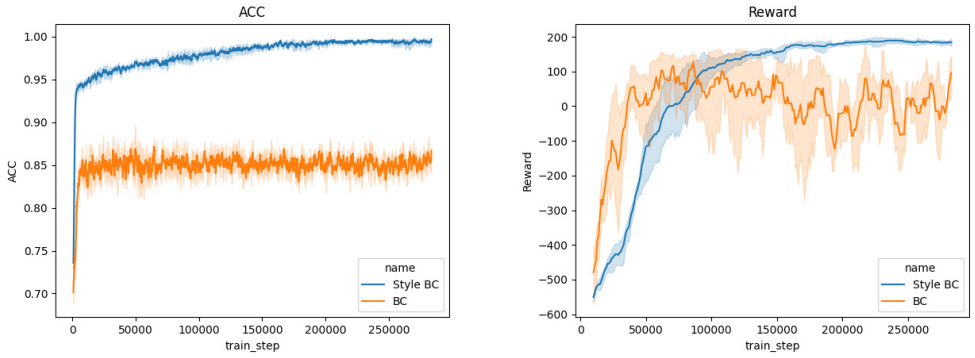


Figure 3: The human demonstrations in the Grid-World environment.



(a) (b)

Figure 4: Experiment results of Style BC and BC in the Grid-world environment. (a) acc and (b) reward.

4.1 GRID-WORLD

4.1.1 EXPERIMENTAL SETUP

We use a 15*15 Grid-World environment as shown in Figure 3, the agent (yellow point) starts from the bottom left corner and is required to arrive at any of the five goal grid (green squares) placed at the right sideline (+20 reward). Besides, there is a wall (black squares) in the vertical direction and the agent can only pass through it from the five doors. We collected trajectories passing through different doors and reaching different five goals, as shown in Figure 3. Based on this, we conduct a set of experiments and intend to verify the following issues: (i) Whether the accuracy of behavior cloning can be improved after introducing style embedding? (ii) Whether the proposed Style BC method can reproduce different style policies? (iii) Is there an exact style embedding vector that can determine a unique style policy?

For the first issue, by comparing Style BC and traditional BC method in the above environment, we observe the differences in accuracy and reward. On this foundation, we introduce two popular imi-

tation learning algorithms, Gail and info-GAIL, and count the number of behavior styles displayed by these four algorithms. As for the last issue, several explicit style embedding is given to learn whether different styles of paths are produced.

4.1.2 EXPERIMENTAL ANALYSIS

First of all, from the results in Figure 4, we can find that the accuracy of Style BC is much higher than traditional BC, which proves that the introduction of style embedding will improve the prediction accuracy of behavior cloning. Moreover, the advantage of adopting style embedding is also reflected in reward. In the later stage of training, the reward of Style BC tends to be stable. On the contrary, the traditional BC's reward fluctuates greatly because this method is sensitive to prediction errors. Figure 5 shows the number of behavior styles displayed by the four algorithms. The demonstration path is a collection of trajectories that pass through five gates and reach different goals, as shown in Figure 3. Therefore, we consider that there are 25 motion strategies in this expert demonstration. Style BC method proposed in this paper reproduces 22 different motion paths, which is the most among the four comparison algorithms; Gail and BC finally converge to only one trajectory, which reflects that the traditional imitation learning algorithm may not be able to show a variety of behavior styles; Info-GAIL represents four different motion trajectories, while we find that some different latent factors correspond to the same behavior style, resulting in only four style paths. In particular, although our method does not completely reproduce the 25 style trajectories, the number of styles reflected is much higher than the four in info-GAIL. Finally, we randomly select three vectors from the style embedding dataset generated by automatic learning as the test style. By interacting with the Grid-world environment respectively, three different motion trajectories can be obtained, as shown in Figure 6. Therefore, under the determined style embedding, only one style of behavior policy can be generated.

4.2 CARTPOLE

4.2.1 EXPERIMENT SETUP

This set of experiments are based on the classic control tasks named CartPole-v1 in OpenAI Gym. In this research, we assume that keeping the pendulum upright and finally staying at different specific positions (0, $\pm$0.5, $\pm$1.5 distance to center in CartPole) represents various behavior styles, shown in Figure 7. The expert demonstrations collect the trajectories where the end point stays at these five points. In the following experiments, Style BC is trained with the demonstrations and is hoped to reproduce different behavior styles. Similarly, we also introduce BC, GAIL and info-GAIL as the comparison algorithms.

4.2.2 EXPERIMENTAL ANALYSIS

The results of CartPole-v1 are demonstrated in Table 1. It's obvious that Style BC and info-GAIL are much higher in style diversity than the other two models. BC learns the behavior style with the end point near -1.5 (-1.46), while algorithm GAIL generates a path with the end stable point near 0 (-0.02). Both behavior styles are from expert demonstrations. Furthermore, we can see that the

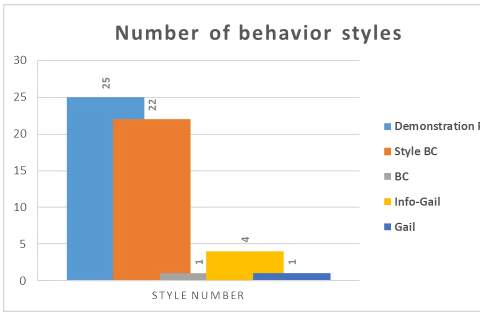


Figure 5: The number of behavior styles.

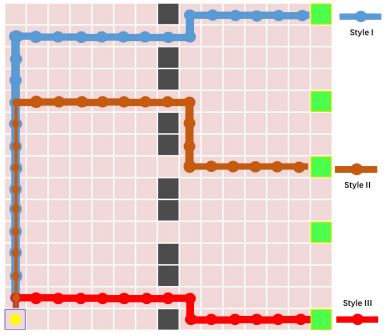


Figure 6: Trajectories of the executed policies.

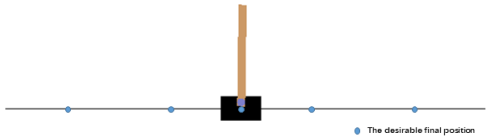


Figure 7: The Cartpole environment.

trajectory end point shown by Style BC is stable at 5 different positons, of which four positions stay in our expected position range. This shows that the Style BC model can replicate the four style representations shown by expert demonstrations. In particular, the end point of one of the tracks generated by Style BC is stable at 0.25. And we believe that the stable point does not fall in the desirable position (0.5) due to the deviation of style embedding. As for Info-GAIL model, although the end of the trajectories it produces also falls in five different positions, these end stable points are inconsistent with the expected position. This does not meet the requirements of reproducing different behavior styles.

4.3 HIGHWAY-ENV

4.3.1 EXPERIMENT SETUP

To demonstrate the effectiveness of Style BC in complex driving environment, the last experiment is carried out on Highway-env, which is a minimalist environment for decision-making in autonomous driving. More details about the environment can be seen at `https://github.com/eleurent/highway-env`.

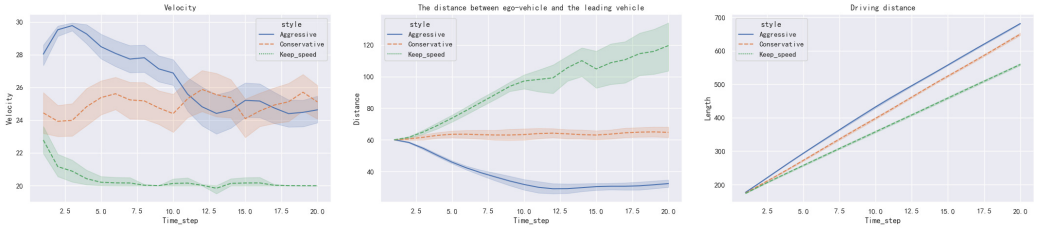
In this experiment, we focus on the style representation of different drivers in the car-following scenario, in which there is no lane-changing behavior. The car-following scenario in the highway-env environment is shown in Figure 1(b). In this environment, the initial position and velocity of surrounding vehicles are random. Moreover, the acceleration of the surrounding agents is given by the Intelligent Driver Model (IDM) (Treiber et al., 2000), and the discrete lane change decisions are given by the Minimizing Overall Braking Induced by Lane change (MOBIL) model (Kesting et al., 2007). Some literatures point out that the traditional car following style can be divided into aggressive and conservative. While in this paper, we also introduce a car-following style, that is, the subject vehicle always keeps the minimum driving speed allowed on the highway (20 m/s) for car-following maneuver. It is assumed that the initial lane of the target vehicle is the middle lane and the initial speed is 25m/s. We collect some demonstration data by a policy trained with PPO and reward shaping that can reflect the above different driving styles. Furthermore, the average velocity, the average distance between the ego-vehicle and the leading vehicle and the total driving distance can be used to identify the different driving styles. Table 2 presents the descriptive statistics of car-following behaviors from the expert demonstrations.

Table 1: Experiment results on Cartpole

ALGORITHM	STYLE NUM	FINAL POS
BC	1	[-1.46]
Style BC	5	[0.48; 1.46; -1.46; 0.25; -0.08]
GAIL	1	[-0.02]
info-GAIL	5	[-2.41; -0.74; -1.38; -0.11; -0.73]

Table 2: Descriptive statistics for different driving styles in car-following manuevers

DRIVING STYLE	AVERAGE VELOCITY	MEAN GAP	TOTAL DISTANCE
Aggressive	$26.29m/s$	$35.05m$	$527.48m$
Conservative	$25.25m/s$	$65.83m$	$505.47m$
Keep_speed	$20.05m/s$	$106.58m$	$403.01m$



(a) (b) (c)

Figure 8: Experiment results on car-following scenario. (a) average speed and (b) mean gap (c) total distance.

4.3.2 Experimental Analysis

We use the Style BC method to learn three style embedding, and each discrete index is used to interact with the highway environment to generate a set of specific style trajectory. Results are provided in Figure 8, where the horizontal axis represents the number of time steps and the vertical axis represents the average speed, mean gap and total distance, respectively. The solid blue line represents aggressive driving style, the dotted red line means conservative style and the green point line expresses maintaining the lowest velocity. In Figure 8(a), the velocity of aggressive driving style fluctuates more violently than the other two models. Accordingly, the distance between the following and the leading vehicle is relatively small in aggressive driving style. The following distance of the conservative driving model is stable at 60m, while the driving total distance of the lowest speed driving style is also the shortest. Furthermore, from Figure 8, we can also find that the three driving styles reproduced are also clearly distinguished in various indicators, which shows that the style embedding can guide the generation of different driving strategies. We recorded the driving process of Style BC with a video which can be found at `https://www.bilibili.com/video/BV1Sf4y1F72P?spm_id_from=444.41.0.0`. During the interaction between the model and the highway environment, these car-following strategies for different driving style can safely complete the whole journey. While the traditional BC, GAIL and info-GAIL methods will collide with surrounding vehicles in the test stage, so they are not included in the comparative experiment.

5 Conclusion

In this paper, we present a method to infer the latent representation of behavior styles automatically while imitating different style policies from expert demonstrations. The novel Style BC method introduces style embedding and proves that it can improve the accuracy of behavior cloning. Our experimental results in some popular benchmark environments show that our methods can reproduce more behavior styles, while an exact style embedding vector can determine a unique style policy. In future work, we seek to explain the performance of different style embedding and find the potential relationship between these style indexes. Besides, extending our method with popular inverse reinforcement learning is also worth researching in the future.

REFERENCES

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Clemens Eppner, Jurgen Sturm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Imitation learning with generalized task descriptions. In *2009 IEEE International Conference on Robotics and Automation*, pp. 3968–3974. IEEE, 2009.

David Sierra González, Jilles Steeve Dibangoye, and Christian Laugier. High-speed highway scene prediction based on driver models learned from demonstrations. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 149–155, 2016. doi: 10.1109/ITSC.2016.7795546.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

Shuhei Ikemoto, Heni Ben Amor, Takashi Minato, Bernhard Jung, and Hiroshi Ishiguro. Physical human-robot interaction: Mutual learning and adaptation. *IEEE robotics & automation magazine*, 19(4):24–35, 2012.

Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

Alex Kuefler and Mykel J Kochenderfer. Burn-in demonstrations for multi-modal imitation learning. *arXiv preprint arXiv:1710.05090*, 2017.

Swagat Kumar. Balancing a cartpole system with reinforcement learning–a tutorial. *arXiv preprint arXiv:2006.04938*, 2020.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3815–3825, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

Jorge Munoz, German Gutierrez, and Araceli Sanchis. Controller for torcs created by imitation. In *2009 IEEE Symposium on Computational Intelligence and Games*, pp. 271–278, 2009. doi: 10.1109/CIG.2009.5286464.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

Cristian Alejandro Vergara Perico, Joris De Schutter, and Erwin Aertbeliën. Combining imitation learning with constraint-based task specification and control. *IEEE Robotics and Automation Letters*, 4(2):1892–1899, 2019.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.