Generating Turn-Based Player Behavior via Experience from Demonstrations

Kuang-Da Wang [1] **Wei-Yao Wang** [1] **Ping-Chun Hsieh** [1] **Wen-Chih Peng** [1]

Abstract

Turn-based sports, such as badminton and tennis, present challenges for imitating human player behaviors from offline datasets in sports analytics. We propose RallyNet, a novel hierarchical offline imitation learning model for turn-based player behaviors. RallyNet captures players' decision dependencies by modeling decision-making processes in turn-based sports as a contextual Markov decision process (CMDP). It leverages experience to generate contexts that aid decision-making, reducing errors. Additionally, RallyNet models player interactions using a latent geometric Brownian motion, enhancing realism and introducing helpful inductive bias. Experimental results on a real-world badminton game dataset demonstrate the effectiveness of RallyNet, outperforming prior offline imitation learning approaches and a state-of-the-art turn-based supervised method.

1. Introduction

Collecting historical data and simulating agents' behaviors have been widely explored to study and replicate specific scenarios in various domains, e.g., autonomous driving. In sports analytics, one of the major goals is to understand and investigate the tactics of teams and individuals. Given the rapid development of information technology, massive amounts of behavioral records in various sports domains can be collected to train models for analysis (Wang et al., 2021; Won et al., 2021; Wang et al., 2022c). If we could create a agent who could recover the player's behavior, these functions would help coaches develop winning strategies, and open up many new applications in players training and sports broadcasting. However, an online environment is unrealistic since it would be impossible to find an opponent to improve the agent's strategies.

[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. Correspondence to: Kuang-Da Wang <gdwang.cs10@nycu.edu.tw>, Wen-Chih Peng <wcpeng@cs.nycu.edu.tw>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling.

Behavior cloning (BC) is one of the offline imitation learning (IL) methods used to recover a player's behavior. The effectiveness of offline IL has recently been illustrated, including hierarchical imitation learning (HIL) models (Wang et al., 2022a; Jing et al., 2021). Although HIL approaches enrich imitated expressivity in the long-horizontal task, none of them were designed for turn-based sports, which consist of multiple players taking actions alternatively to form a rally, meaning that the decision of each player directly affects the decisions of other players. Therefore, there are two challenges to applying existing HIL methods directly for turn-based sports: *1) Leveraging experience.* When players encounter situations that have appeared in their experience, they usually take corresponding actions for returning shots. It is challenging to leverage experience to provide the helpful information of the action for the agent. *2) Alternative decision-making.* In a turn-based sport, the state of each player is determined by the actions of not only themselves but also other players. Thus, the errors of an agent's decision impact the decisions of other agents, resulting in more serious compounding errors.

To address these challenges, we propose a hierarchical offline imitation learning model via experiential context and geometric Brownian motion (RallyNet) to capture long-term decision dependencies by modeling decision-making processes in turn-based sports as contextual Markov decision processes (CMDP) (Hallak et al., 2015). Based on the CMDP setting, the **Experiential Context Selector (ECS)** was designed by establishing the context space from experiences and selecting a context in the space as the agent's intent to mimic the decision-making of the agents following their intents in the rally. This enables the agent's behavior throughout the rally to not be influenced by partially incorrect decisions. Inspired by seeing the interactions between players as being similar to those between particles, we introduce **Latent Geometric Brownian Motion (LGBM)** to capture the interactions between players. We make players alternately complete geometric Brownian motion (Revuz & Yor, 2013) in latent space, which enables the agent's decision-making to jointly consider the opponent's behavior. This generates more realistic behavior in turn-based sports. We highlight our contributions as follows:

- We propose a novel HIL model named RallyNet to mimic player's behavior in turn-based sports.

- We validate RallyNet's performance in real-world badminton, demonstrating its superior final performance.

2. Problem Formulation

To illustrate our approach, we choose badminton as a representative turn-based sport and focus on singles matches involving two players. We denote P_A as the starting player and P_B as the other player for each rally. Let $\mathcal{R} = \{T_r\}_{r=1}^{|\mathcal{R}|}$ denote historical rallies of badminton matches, where the rally is composed of a sequence of state-action pairs. The r-th rally is denoted as $T_r = \{(s_1^{P_A}, a_1^{P_A}), (s_1^{P_B}, a_1^{P_B}), \ldots, (s_{|T_r|}^{p}, a_{|T_r|}^{p})\}$. At the t-th step, s_t^p and a_t^p represent the state and action of the player who takes action, respectively, where $p \in \{P_A, P_B\}$ represents that the step is related to either P_A or P_B. $p = P_A$ when $t = 2i - 1$ and $p = P_B$ when $t = 2i$, where $i = 1, 2, \ldots, |T_r|$. The action consists of the landing positions, shot type, and moving positions represented by $a_t^p = \langle l_t^p, t_t^p, m_t^p \rangle$. The imitation learning task for turn-based sports is to learn a policy that can recover demonstrations from a set of historical rallies $\mathcal{R}$. Formally, for each demonstration rally in $\mathcal{R}$, given initial state s_1^a of the r-th rally T_r, our goal is to recover the rally T_r.

We formulate the decision-making process in badminton games as Contextual Markov Decision Process (CMDP), represented by a tuple (C, S, A, M). Here, C denotes a context space, and M maps each context $c \in C$ with the dimension K to Markov Decision Process (MDP) (Puterman, 2014) parameters[1]. At the beginning of each rally, the agent chooses a context $c \in C$, where c represents the agent's desired intent for the rally. Subsequently, the chosen MDP corresponding to context c is applied throughout the rally until termination. Moreover, we define the *experience* of player p with a current state s_t^p as follows: For a rally that has experienced current states s_t^p, the experience is the action sequence of rallies $\{a_{t|p}^p\}$, where $t|p$ represents the step that is player p taking action. To speed up the extracting experience, we establish an experience extracting function EXP to output the experience from the historical rallies corresponding to the current state; the details can be found in Appendix C.

3. Methodology

Figure 1 illustrates the proposed RallyNet framework, which consists of the following two integral components:

Experiential Context Selector (Section 3.1). We substantiate the idea of leveraging experience via a three-step approach: (i) ECS first extracts experiences by the experience extracting function; and (ii) ECS leverages experiences to construct a latent context space; (iii) ECS selects a context as the agent's selected intent of the rally.

Latent Geometric Brownian Motion (Section 3.2). LGBM effectively brings the inductive bias of Geometric Brownian Motion into the player's decisions and allows the model to consider the behavior of both players jointly instead of separately. Subsequently, the action projection layer (Section 3.3) utilizes the latent position to predict both the landing and moving positions and the shot type for the next step.

Notably, we designed two processes, namely the *target process* and the *player process*. The target process aims to learn to take actions that conform to the context by giving the correct actions as a target context for training. The player process utilizes experiences to learn how to select a context that approaches the output of target processes from the context space. Both processes share most of the modules, and only the player process is used during inferencing. We proceed to present the proposed modules. In this paper, for a player p, we use x_t^p to denote the state embedding for the t-th step of the rally. More details about the state embedding layer (cf. Figure 1) can be found in Appendix B.

3.1. Experiential Context Selector (ECS)

To enable HIL in turn-based sports, we propose the ECS to leverage the experience and provide prior information to the agent, e.g., *what context the rally will have in a current state*. ECS leverages experience to construct a context space in which the agent can select the most relevant context as the intent of the rally, thereby ensuring that the agent's behavior during the rally is not influenced by partially incorrect decisions. We first extract experiences of the current state s_t^p from historical rallies by the extracting function EXP:

$$\{\widetilde{\tau}_t^p, \{\tau_{n,t}^p\}\} \leftarrow EXP(s_t^p, \mathcal{R}), \tag{1}$$

where $\widetilde{\tau}_t^p$ denotes the correct action sequence $\{a_{t|p}^p\}$, and $\{\tau_{n,t}^p\}$ denotes the extracted experiences. For ease of notation, we let $\tau_t^p = \{\widetilde{\tau}_t^p, \{\tau_{n,t}^p\}\}$. We employ the variational autoencoders (Kingma & Welling, 2013; Rezende et al., 2014) architecture as the context encoder and context decoder for learning context representations from experiences. For the t-th step in a rally, the context encoder q encodes experiences $\{\tau_{n,t}^p\}$ into contexts. The context encoder q encodes the n-th experience $\tau_{n,t}^p$ into context z_n:

$$z_n \sim q(\bullet|\tau_{n,t}^p). \tag{2}$$

ECS iteratively encodes every experience into a context and collects them as Z_{ctx} for building a context space. Following existing work (Garnelo et al., 2018) producing latent representations, we average the collected contexts Z_{ctx} to get the centroid of the context space $\bar{z}_{ctx}$:

$$\bar{z}_{ctx} = mean(Z_{ctx}). \tag{3}$$

[1] A MDP is defined by a tuple (S, A, T, R, γ)

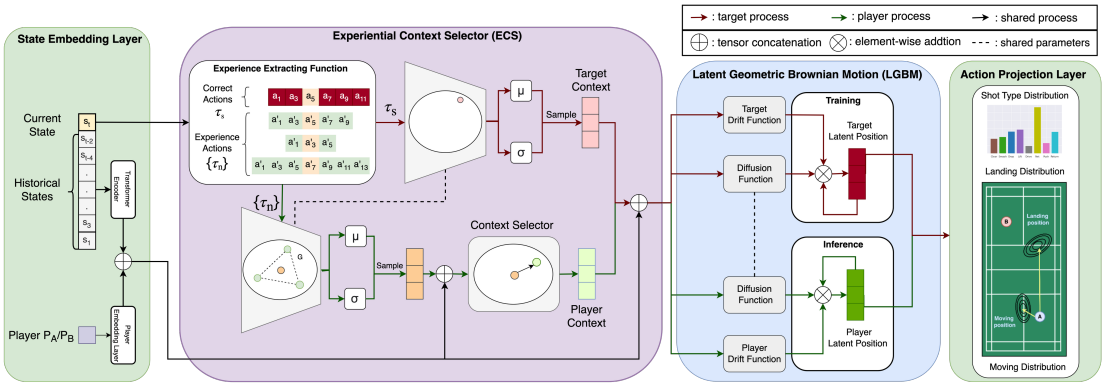

Figure 1. The framework of RallyNet.

We concatenate the centroid of the context space $\bar{z}_{ctx}$ and state embedding x_t^p and use a linear layer as the context selector CTX to select the player context z_{ctx} as intent:

$$z_{ctx} = CTX(concat(\bar{z}_{ctx}, x_t^p)). \tag{4}$$

Since the player process aims to make outputs that are close to the target process, the context selector's goal is to select a player context z_{ctx} that are close to the target context. The target process utilizes the same context encoder q to encode the correct action sequence $\widetilde{\tau}_t^p$ to obtain the target context $\widetilde{z}_{ctx}$:

$$\widetilde{z}_{ctx} \sim q(\bullet|\widetilde{\tau}_t^p). \tag{5}$$

3.2. Latent Geometric Brownian Motion (LGBM)

In turn-based sports, players often consider their opponent's intent and determine their actions; for example, in badminton games, players consider their opponent's intent and next action to determine their defensive position to better receive the shuttlecock. Thus, a player's behavior changes depending on the opponent's intent. To capture player interactions and enhance realism, we propose LGBM. It models players as particles alternately performing geometric Brownian motion in latent space, enabling the agent to jointly, rather than independently, consider the opponent's behavior. Specifically, we first concatenate the agent's state and selected context as the agent's position in the latent space:

$$\widetilde{z}_t' = concat(x_t^p, \widetilde{z}_{ctx}), \quad z_t' = concat(x_t^p, z_{ctx}), \tag{6}$$

where $\widetilde{z}_t'$ is the target latent position for the target process and z_t' is the player latent position for the player process. Secondly, we follow the setting in (Li et al., 2020) to simulate the geometric Brownian motion in the latent space. A Brownian motion is a random process and is described by the following stochastic differential equation (SDE):

$$dX_t = \sigma(x,t)dW_t, \tag{7}$$

where W_t is Brownian motion and σ is the diffusion function. $\Delta W_t \sim \sqrt{\Delta t}\mathcal{N}(0,1)$ is used to simulating the Brownian motion, where $\mathcal{N}(0,1)$ is a normal distribution with zero mean and unit variance. The geometric Brownian motion can be generalized from Eq. (7) with a drift function $h(x,t)$:

$$dX_t = h(x,t)dt + \sigma(x,t)dW_t. \tag{8}$$

We describe the discrete-time SDE of the geometric Brownian motion for two processes, $\Delta\widetilde{z}_t$ and Δz_t as follows:

$$\begin{aligned} \Delta\widetilde{z}_t &= h_\theta(\widetilde{z}_t', t)\Delta t + \sigma(\widetilde{z}_t', t)\Delta W_t, \\ \Delta z_t &= h_\phi(z_t', t)\Delta t + \sigma(z_t', t)\Delta W_t, \end{aligned} \tag{9}$$

where h_θ and h_ϕ are the target drift function and the player drift function, respectively. Both processes share the same function σ. Finally, we add the displacement to the previous latent position to compute the new latent position:

$$z_t = z_{t-1} + \Delta\widetilde{z}_t \ \text{(training)}, \quad z_t = z_{t-1} + \Delta z_t \ \text{(inference)}. \tag{10}$$

LGBM incorporates opponent decisions by considering the displacement, which represents the agent's decision, added to the previous latent position.

3.3. Action Projection Layer

The action projection layer is designed to project the latent position to the action that can interact with the opponent. To predict the shot type, we apply a linear layer to the latent position z_t to predict the shot type $\hat{t}_t^p$ at the t-th step:

$$\hat{t}_t^p = softmax(W^T z_t), \tag{11}$$

where $W^T \in \mathbb{R}^{N_t \times d}$ is a learnable matrix, and N_t is the number of shot types. To predict the landing positions $\hat{l}_t^p$ and the moving positions $\hat{m}_t^p$, we assume the landing distribution and moving distribution are weighted bivariate normal distributions which contain the mean $\{\mu_t\} = \{\langle \mu_x, \mu_y \rangle_t\}$, standard deviation $\{\sigma_t\} = \{\langle \sigma_x, \sigma_y \rangle_t\}$, and weight $\{w_t\}$ of N_g bivariate normal distributions. We apply two linear layers to predict the parameterized distributions for the landing and moving positions:

$$\langle \{\mu_t^{(L,M)}\}, \{\sigma_t^{(L,M)}\}, \{w_t^{(L,M)}\} \rangle = W^{(L,M)} z_t, \tag{12}$$

Table 1. Quantitative results. The best is in boldface and the second best is underlined. Since ShuttleNet predicts based on past information, the symbol − denotes that the result is unavailable.

Model	Task 1			Task 2		
	Land	Shot	Move	Land	Shot	Move
Random agent	1.30	104.3	0.85	1.25	79.25	0.87
Rule-based agent	0.86	61.50	0.56	1.05	58.43	0.63
BC	0.86	55.90	0.44	1.03	54.87	0.57
HBC	<u>0.74</u>	<u>35.53</u>	<u>0.40</u>	0.96	35.00	0.55
ShuttleNet	-	-	-	<u>0.90</u>	<u>34.04</u>	<u>0.50</u>
RallyNet (Ours)	**0.59**	**18.86**	**0.34**	**0.79**	**19.51**	**0.49**

where $W^L \in \mathbb{R}^{5\times N_g\times d}$ and $W^M \in \mathbb{R}^{5\times N_g\times d}$ are two learnable matrices for landing and moving, respectively. Finally, we concatenate the shot types and both landing and moving positions to get the action of agent $\hat{a}_t^p = \langle \hat{t}_t^p, \hat{l}_t^p, \hat{m}_t^p \rangle$ and decide the next state of the opponent.

3.4. Loss Function

To mimic the player's action, we minimize the loss[2]:

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{pred} + w_2 \cdot \mathcal{L}_{ctx} + w_3 \cdot \mathcal{L}_{sde}, \qquad (13)$$

where $w_1, w_2, w_3 \in [0, 1]$ are hyper-parameters to balance the weights of the corresponding losses[3]. We minimize the first loss term $\mathcal{L}_{pred}$, which encompasses cross-entropy loss for shot type prediction and negative log-likelihood losses for landing and moving position prediction. The context encoder encodes each experience and generates the mean and standard deviation for each context. These values are used to sample the context embedding, which is passed to the context decoder for experience reconstruction. Therefore, the second loss term $\mathcal{L}_{ctx}$ consists of the latent loss $\mathcal{L}_{latent}$ and the reconstruction loss $\mathcal{L}_{recon}$. $\mathcal{L}_{latent}$ is the KL divergence between the context distribution and the standard Gaussian distribution $\mathcal{N}(0, 1)$. $\mathcal{L}_{recon}$ is the same as $\mathcal{L}_{pred}$, where we minimize the losses for reconstructed actions. Finally, The player process output needs to be close to the target process as only the player process is used during inferencing. We follow (Li et al., 2020) to minimize the last loss term $\mathcal{L}_{sde}$ between two SDEs in Eq. (9) of LGBM:

$$\mathcal{L}_{sde} = \sum_{T_r\in\{\mathcal{R}\}} \sum_{t=1}^{T} \frac{1}{2} |(h_\phi(\widetilde{z}'_t, t) - h_\theta(\widetilde{z}'_t, t))/\sigma(\widetilde{z}'_t, t)|^2. \quad (14)$$

4. Experiments

4.1. Experimental Setup

Dataset and Baselines. As there is only one public dataset of turn-based sports, we evaluated RallyNet on the badminton singles dataset (Wang et al., 2022c). We compare RallyNet against several baselines, including: 1) Random agent, which samples actions uniformly randomly; 2) Rule-based agent, which samples actions from the extracted experience; 3) Behavior Cloning (BC) (Pomerleau, 1988); 4) Hierarchical Behavioral Cloning (HBC) (Zhang & Paschalidis, 2021), which learns an options-type hierarchical policy from demonstrations; and 5) ShuttleNet (Wang et al., 2022c), which is the state-of-the-art turn-based supervised method that fuses the contexts of rally progress and player styles to predict behavior based on past information. It is noted that ShuttleNet requires at least two steps to encode contexts of the players; therefore, it can't be tested only on the initial state.

Evaluation Metrics. We present two tasks: **Task 1** involves predicting from the initial state only, while **Task 2** requires predicting from the states of the first two steps. Since there is no existing work for imitation of turn-based player behaviors, we propose 3 metrics to measure the similarity between generated rallies and true player rallies, even when their sequence lengths differ. To evaluate the results of shot type prediction, we use Connectionist Temporal Classification (CTC) loss (Graves et al., 2006) for uncertainty measurement, which is defined as the negative log-likelihood of the labels given input sequences. To evaluate the predicted landing and moving positions, we use Dynamic Time Warping (DTW) (Berndt & Clifford, 1994) to calculate the distance between generated sequence and correct sequence.

4.2. Quantitative Results

Table 1 presents the performance of RallyNet and the baselines. We summarize the observations as follows: **Superior Performance of RallyNet.** RallyNet surpasses all the baselines in terms of all metrics, whether given only the initial state, or the state of the first two steps. Note that since players usually have similar positions and shot types for *serving* and *receiving* (i.e., the first two steps), the prediction after two steps is more difficult to predict. These results demonstrate that RallyNet addresses the compounding error in offline imitation learning for turn-based sports well. **Hierarchy Advantage.** We can observe that HBC achieves better prediction than BC, and manifests the importance of using HIL in the turn-based setting. **The importance of leveraging experience.** ShuttleNet performs well compared to other baselines when given the states of the first two steps, as it is specifically designed for turn-based sequences in badminton games. Nonetheless, the comparison of RallyNet and ShuttleNet reveals the importance of leveraging experience in turn-based sports.

4.3. Ablation Studies

As RallyNet can be seen as an extension of BC with the addition of ECS and LGBM, an ablation study was conduc-

[2]The calculation of Eq. 13 are described in Appendix D.2.

[3]The effects of different hyper-parameters and the training and implementation details are described in Appendix D.

ted by developing two variants to investigate the relative contributions of two components: 1) BC w/ ECS, which is RallyNet without the LGBM, 2) Indiv. RallyNet, which is RallyNet with the proposed LGBM replaced by the individual LGBMs (i.e., two independent stochastic processes, one for each player). The result is shown in Table 2.

Table 2. Ablation study of our model.

	Task 1			Task 2		
Model	Land	Shot	Move	Land	Shot	Move
BC	0.86	55.90	0.44	1.03	54.87	0.57
BC w/ ECS	0.68	27.01	0.45	0.92	29.40	0.57
Indiv. RallyNet	0.60	19.06	0.37	0.81	21.37	0.53
RallyNet	**0.59**	**18.86**	**0.34**	**0.79**	**19.51**	**0.49**

The Effect of ECS. Recall that ECS is designed to capture the agent's intent through experience, enabling the agent's behavior throughout the rally to not be influenced by partially incorrect decisions. The agent can already achieve a moderately low prediction error by adding ECS to BC, especially when predicting the shot type and the landing position. However, without the help of LGBM, the agent suffers from a large error in moving position since it disregards the opponent's behavior. **The Effect of LGBM.** Recall that LGBM is meant to jointly capture the interaction of players to help the agent generate more realistic behavior. RallyNet (i.e., BC w/ ECS w/ LGBM) demonstrates a significant improvement in performance, particularly in the prediction of moving positions, compared to the model BC+ECS. Furthermore, we investigated the effect of replacing the proposed LGBM with individual LGBMs (i.e., Indiv. RallyNet). The results showed that without the shared inductive bias introduced by proposed LGBM, the agent ignored players' interaction, resulting in a degradation of the predicted moving position performance. Overall, the results further support the effectiveness of LGBM in RallyNet and provide evidence that RallyNet captures the alternative decision-making nature of the turn-based sports.

5. Case Study: Sports Analytics for Badminton

Simulation of Player Behavior. We describe a use case of the RallyNet in sports analytics for characterizing the player's style by simulating the behavior of the player (e.g., the landing and the subsequent moving position upon a shot) against different opponent players in different scenarios. Figure 2 shows the simulated distributions of the landing and moving position of a player after a *defensive shot*. In this example: (i) This player tends to have a landing position mostly on both sides of the backcourt and only occasionally on the midcourt. (ii) This player tends to move to the midcourt after a defensive shot to prepare for the next stroke. Such characterization can help the coach better understand the player's style and thereby devise tactical plans.

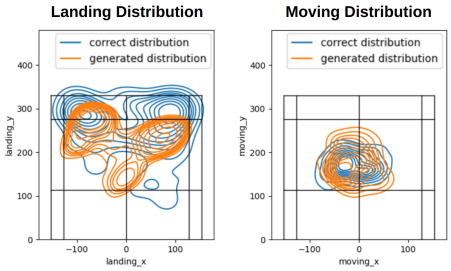

Figure 2. The landing distributions and the moving distributions of a player when using a defensive shot.

Tactical Interpretation of Player Behavior. RallyNet selects a context as the agent's intent, making the agent take action based on the intent, where the intent can be interpreted to obtain the expected action sequence of the agent by the context decoder of ECS. Figure 3 illustrates an example to investigate the underlying intent behind a shot based on the previous shot (long serve in the green grid) and the current shot (net shot in the yellow grid). Player A first chooses the action *clear* with a landing position to the backcourt, and then player A moves to the midcourt. By looking at the decoded intent, we can see the intent behind this action is to plan for a subsequent *smash* by first letting the opponent move far to the backcourt. This example shows that RallyNet can also provide tactical meaning for investigation, which benefits badminton coaches and players.

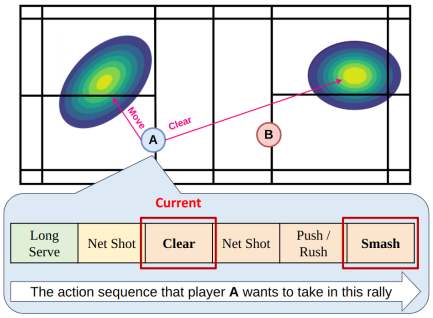

Figure 3. An example of interpreting the intent behind a player's action. Player A's clear stroke is meant to plan for a smash.

6. Conclusion

In this work, we present RallyNet, a novel offline imitation learning model for learning player strategic behavior in turn-based sports. By modeling players' decision-making processes as CMDP, the ECS component reduces impact of partially incorrect decisions, while LGBM captures player interactions to generate more realistic behaviors. Our evaluation of a real-world badminton dataset shows that RallyNet outperforms existing offline IL and the state-of-the-art turn-based supervised method.

Acknowledgements

We sincerely thank all reviewers for their insightful suggestions! This work is partially supported by the Ministry of Science and Technology of Taiwan under Contract No. MOST 112-2425-H-A49-001.

Literatur

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Berndt, D. J. and Clifford, J. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pp. 359–370. Seattle, WA, USA:, 1994.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Garg, D., Chakraborty, S., Cundy, C., Song, J., and Ermon, S. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039, 2021.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International conference on machine learning*, pp. 1704–1713. PMLR, 2018.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Hallak, A., Di Castro, D., and Mannor, S. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Jing, M., Huang, W., Sun, F., Ma, X., Kong, T., Gan, C., and Li, L. Adversarial option-aware hierarchical imitation learning. In *International Conference on Machine Learning*, pp. 5097–5106. PMLR, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Le, H., Jiang, N., Agarwal, A., Dudik, M., Yue, Y., and Daumé III, H. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pp. 2917–2926. PMLR, 2018.

Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. Scalable gradients for stochastic differential equations. *International Conference on Artificial Intelligence and Statistics*, 2020.

Ng, A. Y., Russell, S., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Revuz, D. and Yor, M. *Continuous martingales and Brownian motion*, volume 293. Springer Science & Business Media, 2013.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Schaal, S., Ijspeert, A., and Billard, A. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

Wang, L., Tang, R., He, X., and He, X. Hierarchical imitation learning via subgoal representation learning for dynamic treatment recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1081–1089, 2022a.

Wang, W., Chan, T., Yang, H., Wang, C., Fan, Y., and Peng, W. How is the stroke? inferring shot influence in badminton matches via long short-term dependencies. *ACM Trans. Intell. Syst. Technol.*, 14(1), 2022b.

Wang, W.-Y., Chan, T.-F., Yang, H.-K., Wang, C.-C., Fan, Y.-C., and Peng, W.-C. Exploring the long short-term dependencies to infer shot influence in badminton matches. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 1397–1402. IEEE, 2021.

Wang, W.-Y., Shuai, H.-H., Chang, K.-S., and Peng, W.-C. Shuttlenet: Position-aware fusion of rally progress and player styles for stroke forecasting in badminton. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4219–4227, 2022c.

Won, J., Gopinath, D., and Hodgins, J. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021.

Zhang, Z. and Paschalidis, I. Provable hierarchical imitation learning via em. In *International Conference on Artificial Intelligence and Statistics*, pp. 883–891. PMLR, 2021.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A. RallyNet in a badminton rally

Figure 4 illustrates how RallyNet is applied to a turn-based sport, and uses a badminton singles match as an example. An initial state of the rally including two players' positions and score information is provided to recover the content of the rally. A rally would terminate when any of the following termination conditions are satisfied: (i) The agent misses the shot; (ii) The shuttlecock does not come over the net; (iii) The shuttlecock falls out of the scoring area.

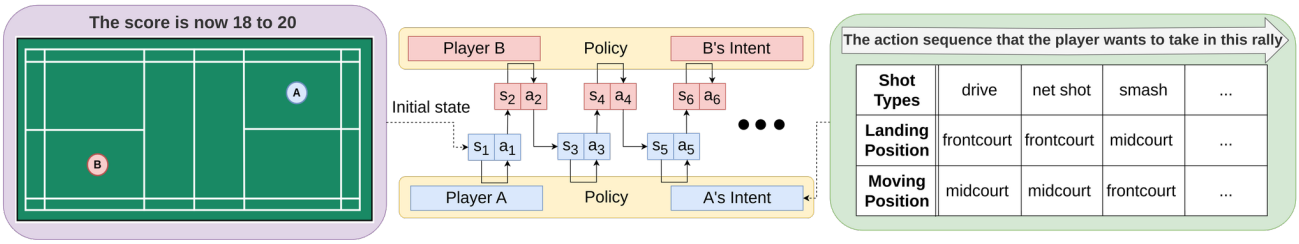

Figure 4. An illustration of RallyNet in a badminton rally.

Limitations Currently, RallyNet focuses on reproducing players' behaviors while does not simulate goal-conditioned behaviors (e.g., provide receipts for winning a game), which will be explored in the future work. Moreover, we plan to extend the framework to badminton doubles with more complex players' behaviors.

B. State Embedding Layer

To capture the long-term decision dependency, we concatenate the current state s_t^p and historical states of the player who currently takes action $s_{1:t-1|p}^p$ and employ a transformer encoder to compute the embedding of the rally r_t^p. To integrate the state with the corresponding player, a linear layer is adopted to compute the embedding of the player who hits the ball $e^p \in \mathbb{R}^{N_p}$ and concatenate with rally embedding $r_t^p \in \mathbb{R}^{d_s}$, where d_s denotes the dimension of states and N_p is the total number of players in the dataset. Formally, the output of state embedding layer at t-th step $x_t^p \in \mathbb{R}^{d_s \times N_p}$ is calculated as follows:

$$\begin{aligned} r_t^p &= TRE(concat(s_t^p, s_{1:t-1|p}^p)), \\ x_t^p &= concat(W^p e^p, r_t^p), \end{aligned} \tag{15}$$

where TRE denotes the transformer encoder and $W^p \in \mathbb{R}^{N_p}$ is a learnable matrix.

C. Experience Extracting Function

The experience extracting function is designed to find historical rallies that are similar to a given input state and to output the corresponding action sequence from those rallies. We create a dictionary before training, where the keys are combinations of discrete values for player positions, opponent positions, ball positions (by discretizing the court into a 10x10 grid), and shot types, and the values are the action sequences of rallies that have experienced combinations of these discrete states. By discretizing the continuous variables and using them as keys in the dictionary, we can efficiently retrieve rallies with matching states.

When a new state is an input into the experience extracting function, we extract the player positions, opponent positions, ball positions, and shot types from the state and retrieve the corresponding rallies from the dictionary. We take the intersection of these rallies to obtain the rallies that have experienced combinations of these discrete states. We then output the action sequences from these rallies, which are used to construct the context space.

Figure 5 illustrates an example of how the experience extracting function works: Assume that the current shuttlecock is at position 25, the player is at position 27, the opponent is at position 35, and the shot type is clear (abbreviated as C). First of all, according to different conditions, find out the experience that meets the conditions. The intersection of all experiences is the experience we want. Since the position-related conditions are strict, if the conditions cannot be met, the condition range

can be expanded. For example: if the opponent is in position 35, but cannot find the experience that meets the conditions, we will expand the range of conditions including 24, 25, 26, 34, 36, 44, 45, 46. If we still can't find the experience that meets the conditions, we continue to expand the scope until we find it.

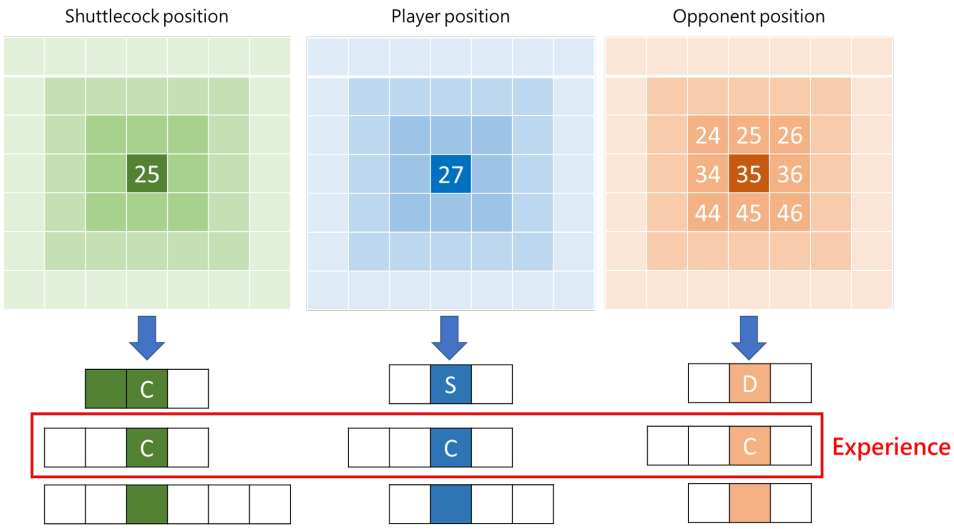

Figure 5. Illustration of experience Extracting function

The use of dictionary retrieval for experience extracting in our approach may introduce some errors, as the selection criterion is relaxed until a proper experience is found. However, the purpose of finding experiences is to establish context spaces for the agent to learn from, and to select the appropriate context. The trade-off between ërroränd "generalization ability"needs to be considered when determining the similarity criterion for experience selection. A stricter similarity criterion can result in finding more similar experiences, and therefore collected contexts can be closer to the target context, reducing the error. However, it is not guaranteed that such similar experiences can always be found during testing, especially in the later stages of a turn-based setting when states become sparse. Therefore, during training, the agent needs to learn how to generalize from experiences to decisions, even when the experiences may not be very similar to the current state. If the similarity criterion is too strict during training, the agent may not be able to learn how to generalize effectively.

D. Experimental Details

In this section, we provide essential experimental details of RallyNet and baselines for reproducibility. All training and evaluation experiments were performed on a machine equipped with an Intel i7-8700 3.2GHz CPU, Nvidia GTX 3060 12GB GPU, and 32GB RAM.

D.1. Dataset

The badminton singles dataset (Wang et al., 2022c) consists of 75 singles matches played by 31 players from 2018 to 2021 and has 180 sets, 4,325 rallies, and 43,191 strokes in total. To construct the action space, we used the 12 shot types defined by (Wang et al., 2022b), namely *receiving*, *short service*, *long service*, *net shot*, *clear*, *push/rush*, *smash*, *defensive shot*, *drive*, *lob, drop*, and *can't reach*.

The range of each dimension of the badminton court was rescaled to [-1,1] to ensure that the model would not be affected by different ranges. We trained our model on the first 80% of the rallies, and the remaining 20% of the rallies were used to evaluate the performance of our model as well as the 5-fold cross-validation for tuning the hyper-parameters.

D.2. RallyNet

Loss function. To mimic the player's action at each step, we minimize the loss:

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{pred} + w_2 \cdot \mathcal{L}_{ctx} + w_3 \cdot \mathcal{L}_{sde}, \tag{16}$$

where w_1, w_2, $w_3 \in [0, 1]$ are hyper-parameters to balance the weights of the corresponding losses.

$$\mathcal{L}_{pred} = \mathcal{L}_{type} + \mathcal{L}_{land} + \mathcal{L}_{move} + \mathcal{L}_{reg}, \quad \mathcal{L}_{type} = -\sum_{T_r \in \{\mathcal{R}\}} \sum_{t=1}^{T} t_t^p \log \hat{t}_t^p, \tag{17}$$

where $\mathcal{L}_{type}$ is the cross-entropy loss for the predicted shot types. $\mathcal{L}_{land}$ and $\mathcal{L}_{move}$ are the negative log-likelihood losses for the prediction of both landing and moving positions. To simplify the expression, we use $\mathcal{L}_{(land,\ move)}$ to represent $\mathcal{L}_{land}$ or $\mathcal{L}_{move}$:

$$\mathcal{L}_{(land,\ move)} = -\sum_{T_r \in \{\mathcal{R}\}} \sum_{t=1}^{T} \log(\mathcal{P}(x_t^{(L,\ M)}, y_t^{(L,\ M)} | \{\mu_t^{(L,\ M)}\}, \{\sigma_t^{(L,\ M)}\}, \{w_t^{(L,\ M)}\})). \tag{18}$$

The regularization loss $\mathcal{L}_{reg}$ is also introduced to prevent the model from degenerating into a simple strategy by ensuring the avoidance of overlapping bivariate normal distributions, computed as the average negative distance between their means.

The context encoder encodes each experience and generates the mean and standard deviation for each context. These values are used to sample the context embedding, which is then passed to the context decoder for experience reconstruction. Therefore, $\mathcal{L}_{ctx}$ consists of the latent loss $\mathcal{L}_{latent}$ and the reconstruction loss $\mathcal{L}_{recon}$. The latent loss $\mathcal{L}_{latent}$ is the KL divergence between the context space distribution and the standard Gaussian distribution $\mathcal{N}(0, 1)$ (with zero mean and unit variance). Specifically, $\mathcal{L}_{latent}$ can be expressed as:

$$\mathcal{L}_{latent} = D_{KL}(\mathcal{N}(\mu_c, \sigma_c) || \mathcal{N}(0, 1)), \tag{19}$$

where μ_c and σ_c are the mean and standard deviation output by the context encoder, respectively. The objective of context decoders is to reconstruct intent, which are action sequences of rallies, from selected contexts. Therefore, the reconstruction loss $\mathcal{L}_{recon}$ is the same as $\mathcal{L}_{pred}$, where we minimize the cross-entropy loss for the reconstructed shot types, the negative log-likelihood losses for the reconstructed landing and moving positions, and the average negative distance between the means of bivariate normal distributions.

The player process output needs to be close to the target process as only the player process is used during inferencing. We follow (Li et al., 2020) to minimize the KL divergence $\mathcal{L}_{sde}$ between two SDEs in Eq. (9) of LGBM:

$$\mathcal{L}_{sde} = \sum_{T_r \in \{\mathcal{R}\}} \sum_{t=1}^{T} \frac{1}{2} |(h_\phi(\tilde{z}_t', t) - h_\theta(\tilde{z}_t', t))/\sigma(\tilde{z}_t', t)|^2. \tag{20}$$

We summarize the hyper-parameters used in the evaluation as follows.

Loss weights. Recall that the loss that we are minimizing is:

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{pred} + w_2 \cdot \mathcal{L}_{ctx} + w_3 \cdot \mathcal{L}_{sde}, \tag{21}$$

where w_1, w_2, $w_3 \in [0, 1]$ are hyper-parameters to balance the weights of the corresponding losses. We have set the loss weights $w1, w2, w3$ to 1. Considering the significant scale difference between L_{land}, L_{move}, and L_{reg} compared to L_{type}, we scale down L_{land}, L_{move}, and L_{reg} by a factor of 0.01. The purpose of L_{reg} is to add a regularization loss that prevents overlap between the predicted landing distributions and moving distributions, ensuring that the model does not degenerate into a simple policy. This regularization loss is defined as the average distance between bivariate normal distributions. The relationship between L_{land}, L_{move}, and L_{reg} is given by:

$$\mathcal{L}(land, move) = (1 - \alpha) \cdot \mathcal{L}(land, move) + \alpha \cdot \mathcal{L}_{sde}, \tag{22}$$

where the regularization loss weight α is set to 0.05 for all experiments. To investigate the influence of varying loss weights on the model's performance, we conducted a parameter analysis by examining different values for w_1, w_2, and w_3.

To facilitate comparison and evaluation, we further provide an overall comparison of the algorithms by introducing a metric termed Rule-based agent Normalized Score (RNS) defined as $RNS = \frac{(Random_{score} - Agent_{score})}{(Random_{score} - Rule_{score})}$, where the subscript $score$ indicates the metric used and can be either DTW distance or CTC loss, and $Agent_{score}$ is the performance of the agent. $Random_{score}$ and $Rule_{score}$ are the performance of the random agent and the rule-based agent, respectively. To provide an overall comparison of the algorithms, we also present the Mean RNS (MRNS), which is defined as the average over the

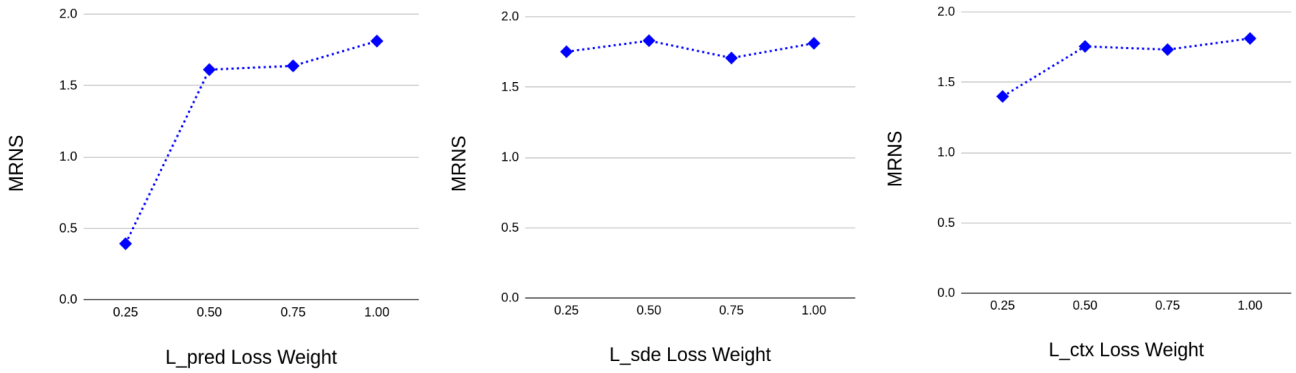

Figure 6. MRNS Performance of RallyNet w.r.t different value of loss weights.

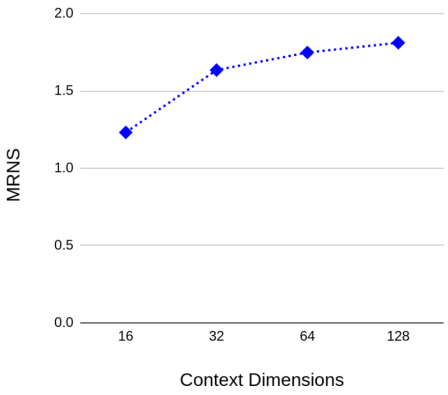

Figure 7. MRNS Performance of RallyNet w.r.t different value of context dimensions.

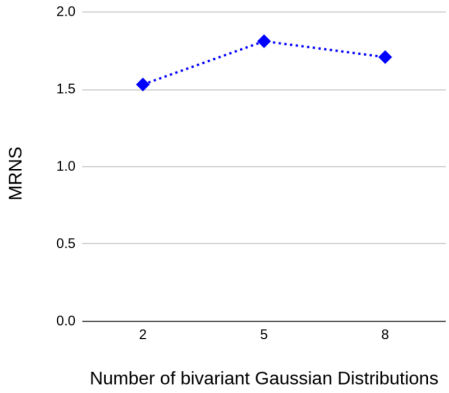

Figure 8. MRNS Performance of RallyNet w.r.t different number of distributions

RNS values under the three base metrics. Therefore, we evaluated the model's MRNS using weights of 0.25, 0.50, and 0.75 for each loss term. The results of these experiments are presented in Figure 6.

The results indicate that increasing the weights of $\mathcal{L}_{pred}$ and $\mathcal{L}_{ctx}$ improves performance by enhancing the accuracy of target process outputs and context representation learning. This enables the player process used for inference to closely approximate the precise target process. We observed that the influence of L_{sde} on overall performance is stable. This suggests that when the target process is effectively learned, the player process effortlessly approximates the target process, resulting in consistent performance that is minimally affected by L_{sde}.

Conetext dimension. In all experiments, we set the context embedding dimension to 128. The dimension of the context embedding plays a crucial role in determining the continuity of the context space and influencing the specificity of the selected context (i.e., player's intention). As shown in Figure 7, increasing the dimension of the context embedding results in improved overall performance.

The number of bivariate Gaussian distributions. We conducted experiments with three different numbers of bivariate Gaussian distributions: 2, 5, and 8. Figure 8 reveals that insufficient or excessive distributions were unable to adequately capture the intricate distributions of landing positions and movement locations.

Implementation details of predicted positions. To avoid the predictions for landing positions and movement positions encompassing the entire court and degenerating the strategy, we imposed a constraint on the standard deviation of the bivariate Gaussian distributions. The standard deviation was limited to the range of [-0.1, 0.1].

Implementation details of experience extracting function. To improve the model's generalization capability and mitigate over-reliance on experiences for decision-making, we utilized an experience extracting function that restricted the number of experiences for the current state to 5. This was done because a substantial amount of experience cannot be guaranteed during testing, particularly in the later stages of a turn-based setting when the states become sparse.

The additional hyperparameters are listed in Table 3.

Table 3. Hyperparameters for RallyNet.

Hyperparameter	Value
State embedding dimension	80
Batch size	32
Learning rate	0.001
Dropout	0.1
Max epochs	50

D.3. Baselines

We treated the number of options for HBC as a hyperparameter and conducted experiments using various numbers of options $\{2, 4, 8\}$. Among these, the best result was achieved when the number of options was set to 8. Since ShuttleNet originally predicted only the shot type and landing positions, we extended its capabilities by incorporating a module similar to the landing position prediction to also enable the prediction of moving positions.

E. Related Work

Inverse Reinforcement Learning. Inverse Reinforcement Learning (IRL) (Ng et al., 2000; Abbeel & Ng, 2004; Fu et al., 2017) is an imitation learning approach that attempts to learn the underlying rewards that an expert is optimizing from the expert demonstrations. With the learned reward function, IRL can guide the agent's behavior toward that of the expert. However, in turn-based sports, each player's actions determine the next state for other players. Treating opponents as part of the environment is equivalent to training in a constantly changing environment, which can make it challenging for IRL to learn optimal policies.

Offline Imitation Learning via Behavior Cloning. Behavior cloning (BC) provides a form of supervised learning for training policies by learning direct mapping from states to actions, which can be used in the offline setting (Pomerleau, 1988; Schaal et al., 2003). Recent works such as DT (Chen et al., 2021), GCSL (Ghosh et al., 2019), and RvS (Emmons et al., 2021) have shown that not only is supervised learning and bypassing the learning reward function able to attain better results for offline learning, but it is also easier to use and is more stable than offline inverse reinforcement learning (IRL) (Ng et al., 2000; Ziebart et al., 2008; Garg et al., 2021). However, these existing approaches focus on the imitation learning task of the same target, but neglect the characteristics of the mixed sequences, which causes serious compounding errors. Moreover, previous work (Le et al., 2018; Zhang & Paschalidis, 2021) has extended the BC to hierarchical BC by dividing a task into several sub-tasks, where the low-level behaviors are controlled by high-level decisions to capture long-term decision-making processes. However, these existing approaches focus on the complex behavior and the long-term task of the same target, but neglect the characteristics of the mixed sequences, which causes serious compounding errors.