Exploiting Geometric Structures for Modeling Multi-Agent Behaviors: A New Thinking

Bohao Qu[1], Xiaofeng Cao[2*], Bing Li[1], Menglin Zhang[3], Tuan-Anh Vu[4], Di Lin[5], Qing Guo[6*]

[1] CFAR and IHPC, Agency for Science, Technology and Research (A*STAR), Singapore,
[2] School of Computer Science and Technology, Tongji University, China,
[3] College of Computer Science and Technology, Jilin University, China,
[4] Department of Mechanical & Aerospace Engineering, University of California, Los Angeles (UCLA), USA,
[5] College of Intelligence and Computing, Tianjin University, China,
[6] VCIP, School of Computer Science, Nankai University, China.

Abstract

In this paper, we rethink model agent behaviors from a geometric structure perspective in multi-agent reinforcement learning. Modeling agent behaviors is essential for understanding how agents interact and facilitating effective decisions. The key lies in capturing the dependencies and sequential relationships among agent decisions. Since each decision influences the subsequent choices, this forms a hierarchical and nested tree-like structure of interdependencies. While modeling tree-like data in Euclidean spaces could cause distortion, which results in a loss of agent decision structure information. Motivated by this, we reconsider model agent behaviors in hyperbolic space, and propose the **H**yperbolic **M**ulti-**A**gent **R**epresentations (HMAR) method, which projects the agent behaviors into a Poincaré ball and leverages hyperbolic neural networks to learn agent policy representations. Additionally, we designed a contrastive loss function to train this network, minimizing the distance in feature space between different representations of the same agent while maximizing the distance between representations of distinct agents. Experimental results provide empirical evidence for the effectiveness of the HMAR method in cooperative and competitive environments, demonstrating the potential of hyperbolic agent representations for effective decision-making in multi-agent environments.

Introduction

Recently, the study of modeling agent behaviors in complex and dynamic multi-agent environments has garnered significant attention, particularly within the realm of multi-agent reinforcement learning (MARL) (Albrecht and Stone 2018; He et al. 2016; Rabinowitz et al. 2018; Grover et al. 2018; Raileanu et al. 2018; Papoudakis, Christianos, and Albrecht 2021). As multi-agents become increasingly prevalent in various domains, from robotics (Luo, Ni, and Tian 2022; Dasari et al. 2020) and autonomous driving (Natan and Miura 2022; Zhou et al. 2021) to financial trading (Sun et al. 2024; Fang et al. 2023) and strategic games (Silver et al. 2017; Vinyals et al. 2019; Silver et al. 2018), understanding and modeling their interactions have become crucial for enhancing their decision-making capabilities.

*Corresponding Authors.

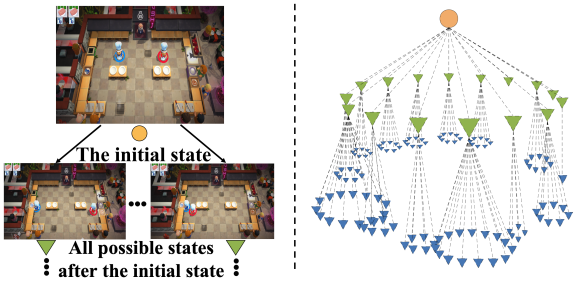

Figure 1: In the left subfigure, the yellow dot represents the initial state of *Overcooked*, with green triangles showing the subsequent states. The right subfigure abstracts this process as a tree structure from the perspective of an agent, and blue triangles represent successor states to the green ones.

Modeling agent behaviors is fundamentally about learning representations that accurately reflect the interactions between agents. These representations are essential as they enable agents to predict and respond to the actions of others, thus facilitating more effective and coordinated decision-making processes. The key to learning agent representations lies in capturing the dependencies and sequential relationships inherent in agent decisions. Duo to each agent's decision engenders a spectrum of potential future states, and the decision is also influenced by prior states, this process creates a hierarchical and nested structure of interdependencies.

Structured new thinking. *Based on the structural characteristics of multi-agent decision processes, we consider geometric objects that are isomorphic to them. This multiplicity of decisions manifests as a tree-like structure (Cetin et al. 2024; Dong, Zhang, and Zhang 2023), where each node represents a decision that bifurcates into branches. These branches symbolize the disparate actions undertaken by the agents. Sequentially, each level of the tree corresponds to a time step, and decisions are arranged in a hierarchy of parent-child relationships. As the number of agents increases, the number of feasible future states grows exponentially, and both the depth and breadth of the tree increase accordingly. We show the tree-like structure of the decisions from the per-*

spective of an agent in Figure 1.

Challenges in new thinking. *This tree-like structure poses a significant challenge for traditional Euclidean space-based methods, which struggle to represent such data accurately due to their polynomial space growth properties (Chami et al. 2019; Sarkar 2011; Sala et al. 2018). Consequently, this results in substantial distortion and a severe loss of crucial agent decision structure information.*

Motivated by this, we reconsider pivot to modeling agent behaviors within hyperbolic space, a realm ideally suited to representing hierarchical features with low distortion by the exponential space growth properties (Beltrami 1868b; Cannon et al. 1997; Sarkar 2011; Sala et al. 2018). Embedding the tree structure in hyperbolic space allows for a homeomorphic embedded, capturing the decision-making dynamics and state evolution topology in multi-agent sequential decisions. This not only preserves characteristic relationships (e.g. the policy's preference for certain actions in specific situations) inherent in the original Euclidean framework but also aptly mirrors the hierarchical progression of multi-agent decision routes across timesteps.

Based on these considerations, we propose a novel method **H**yperbolic **M**ulti-**A**gent **R**epresentations (HMAR), which projects the trajectories into a Poincaré ball and leverages the hyperbolic neural networks to learn agent policy representations, preserving the inherent hierarchical structures with minimal distortion. The Poincaré ball is a suitable choice for our setting, since the distance within the Poincaré ball changes smoothly, and this locality property of the distance is key for learning continuous embeddings of hierarchies (Nickel and Kiela 2018). We also devise a contrastive loss function for training this network, which minimizes the distance in feature space between different representations of the same agent, enhancing self-behavior understanding, while simultaneously maximizing the distance between representations of distinct agents, facilitating better comprehension of other agents' behaviors. The resultant agent representations not only condition the policies but are also concurrently optimized with the agent policy model during the reinforcement learning (RL) training process.

We evaluate the performance of HMAR in two multi-agent environments (one cooperative, one competitive): *Overcooked* (Carroll et al. 2019), and *Pommerman* (Resnick et al. 2018). The experimental results support the idea that effective agent modeling can be achieved in hyperbolic space. The same RL algorithm generally achieved higher average returns when combined with representations generated by HMAR than without, and these remarkable findings emphasize the tremendous potential of Poincaré agent representations and illuminate the path to enhancing policy optimization in multi-agent environments.

Related Work

Learning agent models. Agent modeling is crucial to understanding the emergence of complex phenomena of agents in MARL. A generative method is proposed in (Grover et al. 2018; Zhang et al. 2025), which proposed an innovative approach that relies on imitation learning where they train a mapping from observations to actions in a supervised manner to capture a point-based policy representation. Three meta-learning methods are proposed in (Rusu et al. 2018; Ghosh and Bellemare 2020; Qu et al. 2024), and they regard the latent generative representation of learning model parameters as the policy representation. (Tacchetti et al. 2018) introduced a notable concept involving relational forward models that utilize graph neural networks for modeling agents. (Zintgraf et al. 2021) employed a Variational Autoencoder (VAE) for agent modeling, particularly for fully-observable tasks. (Rabinowitz et al. 2018) proposed the Theory of mind Network (TomNet), which learns embedding-based representations of modeled agents for meta-learning. (Cetin et al. 2024) shows that performance improvements of RL algorithms correlate with the increasing hyperbolicity of the discrete space spanned by their latent representations and also validates the importance of appropriately encoding hierarchical information. (He et al. 2016) introduced a novel method that focuses on the learning of a modeling network tasked with reconstructing the actions of a modeled agent based on its observations. (Raileanu et al. 2018) contributed to the field by developing an algorithm designed to learn the inference of an agent's intentions by leveraging the policy of the controlled agent. Prior works assumed that the behavior has Euclidean property with state linear transformation, we rethink that behavior has an implicit hierarchical property, which induces a tree-like structure.

Hyperbolic representations. Hyperbolic geometry introduced by Beltrami (Beltrami 1868a) and further developed by Cannon (Cannon et al. 1997), offers a compelling framework for modeling hierarchically structured features and capturing non-linear relationships, exemplified by the Poincaré ball. Hyperbolic space has found application in various embedding tasks (Kleinberg 2007; Walter 2004; Shavitt and Tankel 2008; Krioukov et al. 2009; Cvetkovski and Crovella 2009; Krioukov et al. 2010; Bläsius et al. 2018). Building upon this foundation, Nickel et al. (Nickel and Kiela 2017) proposed learning Poincaré embeddings for symbolic data, emphasizing latent hierarchical structures. Dhingra et al. (Dhingra et al. 2018) extended these ideas to text and sentence embedding using the Poincaré model, eliminating the need for a projection step through re-parametrization. Tifrea et al. (Tifrea, Bécigneul, and Ganea 2018) innovatively embedded words within a Cartesian product of hyperbolic spaces, drawing theoretical connections to Gaussian word embeddings and Fisher geometry. Addressing numerical instability concerns, Yu et al. (Yu and De Sa 2019) developed a method capable of representing any point with a small, fixed bounded error within hyperbolic networks. Despite the growing popularity of hyperbolic representations in machine learning, there have not been notable extensions for MARL (Peng et al. 2021).

Problem Formulation

In MARL, the primary objective for each agent is to learn a policy that effectively coordinates or competes with others in the environment. In this work, we assume each agent operates with a single policy. This policy dictates how the agent perceives its environment and makes decisions. We model the problem with the classical framework of the Markov Game

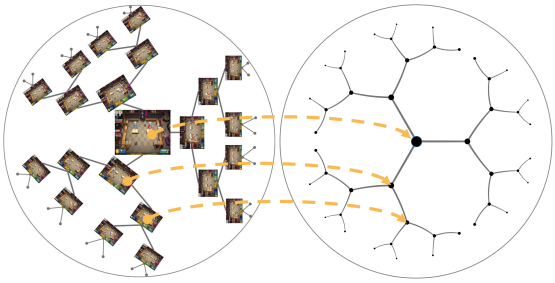

Figure 2: The left figure shows a tree-like structure of states formed over timesteps in *Overcooked*, which can be homomorphically embedded into a hyperbolic space depicted in the right figure. The dashed lines are the corresponding nodes.

(Littman 1994). This framework encompasses a set of N agents, denoted as $\mathbb{I} = 1, 2, ..., N$, operating within a state space $\mathcal{S}$. The agents interact through a joint action space $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_N$. The dynamics of the environment are governed by a transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, which determines the state transitions based on the collective actions of the agents. Additionally, for each agent $i \in \mathbb{I}$, there is a designated reward function $R_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, guiding their individual decision-making processes.

The agent i operates under policy π_{θ_i}, with parameters θ_i, to choose its actions. Interactions between the agent i and its environment yield trajectories $\tau_i = \{(s_i^t, a_i^t)\}_{t=1}^T$, encapsulating state-action sequences over time, with t and T denotes the current and maximum time step respectively. The goal of the agent i is to learn a policy that maximizes its expected discounted accumulative rewards,

$$\arg\max_{\theta_i} \mathbb{E}_{\tau_i \sim \pi_{\theta_i}, P} \left[\sum_{t=0}^{\infty} \gamma^t R_i(s_i^t, a_i^t) \right], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor that quantifies the agent's inclination towards earlier rewards.

Using trajectories to learn agent representations is an effective approach (Grover et al. 2018; Papoudakis, Christianos, and Albrecht 2021). Based on the tree-like structure of trajectories, we embed them homomorphically into a Poincaré ball. Specifically, we conceptualize the trajectories as a tree's growth process, with the initial state of each trajectory positioned at the ball's center. As agents interact with the environment, this 'tree' expands from the central region towards the boundary of the ball. Figure 2 (left) illustrates the tree-like structure composed of states from multiple timesteps displayed on a plane. Correspondingly, hyperbolic space, depicted in Figure 2 (right), can be viewed as a continuous analog of trees.

Our objective is to optimize the representations for each agent, which are utilized to condition the agent policy. We transform the K trajectories $E_i = \{\tau_i^k\}_{k=1}^K$ of each agent i to a agent representation set $\{\hat{\pi}_i\}_{k=1}^K$, situated within the m-dimensional Poincaré ball $\mathbb{B}_c^m$ of a constant negative curvature $-c$. This transformation is defined by the representation function $f_{\boldsymbol{\Theta}} : E_i \to \{\hat{\pi}_i\}_{k=1}^K \in \mathbb{B}_c^m$, and the function's parameters $\boldsymbol{\Theta} = \Theta_1, \ldots, \Theta_L$ are organized layer-wise for deep learning architectures. Each layer Θ_l comprises both weight parameters Θ_l^w and bias parameters Θ_l^b.

Methodology

Hyperbolic representation generation module

Project trajectories into Poincaré ball. We propose a novel approach to generate agent policy representations by embedding agent trajectories within the Poincaré ball, aiming to preserve the intrinsic hierarchical structure with minimal distortion. The process begins by projecting the trajectory space $E_i = \{\tau_i^k\}_{k=1}^K$ of agent i into the Poincaré ball $\mathbb{B}_c^m$. We employ the exponential map for this projection, represented as follows:

$$\hat{\tau}_i^k = \exp_{\mathbf{0}}^c(\tau_i^k) = \tanh\left(\sqrt{c}\|\tau_i^k\|\right) \frac{\tau_i^k}{\sqrt{c}\|\tau_i^k\|}, \tag{2}$$

where τ_i^k is the k trajectory of agent i, and $\hat{\tau}_i^k$ denotes the result of τ_i^k mapped into the Poincaré ball.

Hyperbolic neural network for multi-agent representations. To extract hierarchical relationships and other features from the trajectories, we utilize the hyperbolic neural network (Grover et al. 2018; Papoudakis, Christianos, and Albrecht 2021), which is designed to capitalize on the high capacity and tree-like structures inherent to hyperbolic space. The neural network operations are defined using Möbius matrix-vector multiplication (Ganea, Bécigneul, and Hofmann 2018), particularly for each layer l with weight parameters Θ_l^w. For a trajectory $\hat{\tau}_i^k \in \mathbb{B}_c^m$, the operation is:

$$\Theta_l^w \otimes_c \hat{\tau}_i^k = \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\Theta_l^w \hat{\tau}_i^k\|}{\|\hat{\tau}_i^k\|} \tanh^{-1}(\sqrt{c}\|\hat{\tau}_i^k\|) \right) \frac{\Theta_l^w \hat{\tau}_i^k}{\|\Theta_l^w \hat{\tau}_i^k\|}. \tag{3}$$

Bias adjustments in the neural network are achieved by moving along geodesics within the Poincaré ball, implemented through parallel transport:

$$\begin{aligned} \hat{\tau}_i^k \oplus_c \Theta_l^b &= \exp_{\hat{\tau}_i^k}^c(P_{\mathbf{0} \to \hat{\tau}_i^k}^c(\log_{\mathbf{0}}^c(\Theta_l^b))) \\ &= \exp_{\hat{\tau}_i^k}^c\left(\frac{\lambda_{\mathbf{0}}^c}{\lambda_{\hat{\tau}_i^k}^c} \log_{\mathbf{0}}^c(\Theta_l^b) \right). \end{aligned} \tag{4}$$

The synthesis of these equations forms the layer l of the hyperbolic neural network,

$$f_{\Theta_l}(\tau_i^k) = \Theta_l^w \otimes_c \left(\exp_{\mathbf{0}}^c\left(\tau_i^k\right)\right) \oplus_c \Theta_l^b. \tag{5}$$

We get the k-th representation of agent i by incorporating multiple layers denotes as $\hat{\pi}_i^k = f_{\boldsymbol{\Theta}}(\tau_i^k)$. This formulation not only captures the complex structural and hierarchical nuances within the trajectories but also ensures minimal distortion in representing the structures. We generate multiple representations for each agent. The representations of multi-agents are used to train this module.

Optimize representation module for multi-agent

Consistent multi-agent representations. For multi-agent systems, the representations derived from different trajectories of the same agent must maintain consistency, signifying the adherence to common policy characteristics, and should exhibit proximity in the embedding space.

The trajectories generated by an agent may exhibit substantial variability, which arises due to several factors, different starting states or conditions can lead to diverse trajectories under the same agent policy, and in MARL, the presence of other agents, each following their policy, adds complexity and unpredictability to the trajectories of any agent policy.

Despite this variability, these trajectories can still reveal characteristics of the agent policy. Repeated patterns in decision-making, such as preference for certain actions in specific situations, become evident across different trajectories. The agent policy might consistently demonstrate a tendency towards risk-averse or risk-seeking behavior, which can be observed across various trajectories. The agent policy might prioritize long-term rewards over immediate gains or vice versa, influencing the trajectory patterns. While the trajectories generated by a single policy can be diverse due to various factors, they collectively offer a show of the agent policy's unique characteristics.

Considering that the agent policy embeddings generated by the hyperbolic neural network remain within the Poincaré ball, the consistency is measured using the *geodesic distance* in the Poincaré ball:

$$d_{\mathbb{B}}(\hat{\pi}_i^k, \hat{\pi}_i^{k'}) = cosh^{-1}\left(1 + 2\frac{\|\hat{\pi}_i^k - \hat{\pi}_i^{k'}\|^2}{(1 - \|\hat{\pi}_i^k\|^2)(1 - \|\hat{\pi}_i^{k'}\|^2)}\right), \tag{6}$$

where $1 \leq k, k' \leq K$, k and k' represent the different trajectories of agent i. The consistency objective function $\mathcal{L}_{con}(\mathbf{\Theta})$ is then defined to minimize distances within the same policy embeddings:

$$\mathcal{L}_{con}(\mathbf{\Theta}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{k\neq k'}\left[\begin{array}{l}\log\left(1 + \dfrac{d_{\mathbb{B}}\left(\hat{\pi}_i^k, \hat{\pi}_i^{k'}\right)}{\epsilon}\right) \\ + \mu\left(\log\dfrac{1}{1 - \|\hat{\pi}_i^k\|}\right)\end{array}\right], \tag{7}$$

where $\hat{\pi}_i^k = f_{\mathbf{\Theta}}(\tau_i^k)$, $\hat{\pi}_i^{k'} = f_{\mathbf{\Theta}}(\tau_i^{k'})$, N denotes the number of agents, ϵ serves as an adjustment parameter, and μ is the regularization coefficient. Employing the logarithm effectively scales down the calculated distances and norms within the embeddings in the Poincaré ball, resulting in smoother data without altering the fundamental nature of the data or its relationships. The term on the left side of the plus sign is designed to minimize the distance in the Poincaré ball between embeddings. The denominator includes a tuning parameter ϵ to adjust the distance between the two embeddings according to different tasks. The term on the right side of the plus sign aims to minimize the embedding within the bounds of the radius in the Poincaré ball. Placing the norm in the denominator ensures that the result remains positive when taking the logarithm. By minimizing the consistent objective function, distinct embeddings of the same policy can cluster within the embedding space.

Discriminative multi-agent representations. Different agents inherently exhibit diverse action distributions given the same state, leading to distinct characteristics. Discrimination stems from the varying action preferences exhibited by different agents, leading to distinctive characteristics in their respective trajectories during interactions with both the environment and other agents. Hence, it becomes imperative for embeddings to clearly portray these distinctions among different agents.

In MARL, it's possible for agents to interact with the environment with policies that appear similar. However, due to the complex interplay of factors in such environments, even minor differences in policies can lead to significantly different behaviors over time. Each agent may start from different states or encounter unique sequences of events, shaping their policy in subtly different ways. Agents learn and adapt based on their interactions with the environment and other agents. This process is inherently individualistic, leading to unique policy evolution. Random elements in the environment or in the agents' decision-making processes can lead to divergence in policies that initially seem similar. On the other side, the multi-agent hyperbolic representation module captures a wide range of features and nuances of policies. Subtle variations in agent policies, such as slightly different probabilities of taking certain actions in specific states, become distinguishable in the embedding space.

To accentuate the differences between distinct agent policies, embeddings should exhibit clear boundaries. The discriminative objective function $\mathcal{L}_{dis}(\mathbf{\Theta})$ maximizes distances between embeddings of different agent policies:

$$\mathcal{L}_{dis}(\mathbf{\Theta}) = \frac{1}{N}\sum_{i\neq i'}^{N}\mathbb{E}_k\left[\log\left(1 + \frac{d_{\mathbb{B}}\left(\hat{\pi}_i^k, \hat{\pi}_{i'}^k\right)}{\epsilon}\right)\right], \tag{8}$$

where $\hat{\pi}_i^k = f_{\mathbf{\Theta}}(\tau_i^k)$, $\hat{\pi}_i^{k'} = f_{\mathbf{\Theta}}(\tau_i^{k'})$, N and ϵ also denote the number of agents and an adjustment parameter, respectively. Within Equation (8), the application of the logarithm operation and the incorporation of ϵ in the denominator follow the identical rationale as delineated in Equation (7).

Ensemble optimization. The ensemble objective $\mathcal{L}_{ens}$ integrates the consistency and discriminative objectives, utilizing a trade-off parameter β:

$$\mathcal{L}_{ens} = \mathcal{L}_{con} - \beta\mathcal{L}_{dis}. \tag{9}$$

Optimization is achieved through stochastic Riemannian gradient descent (RSGD) (Bonnabel 2013), balancing between capturing agent policy characteristics and differentiating between distinct agents. The algorithm for the proposed HMAR method is presented in the Appendix.

Multi-agent reinforcement learning training

The agent representations are designed to serve MARL, providing auxiliary information for agent decision-making in cooperative or competitive settings. The Poincaré agent policy embeddings can be used to condition the RL policies. That is,

the input to the actor and critic of the RL algorithm is the state and the agent policy embedding. We do not back-propagate the gradient from the actor-critic loss to the parameters of the multi-agent hyperbolic representation module. We use different learning rates for optimizing the parameters of the policy networks of RL and the multi-agent hyperbolic representation module. Based on the general RL objective Equation (1), we use the agent policy embeddings as the policy condition to optimal policy for each agent i:

$$\arg\max_{\theta_i} \mathbb{E}_{\tau_i \sim \pi_{\theta_i}, P} \left[\sum_{t=0}^{\infty} \gamma^t R_i((s_i^t, \hat{\pi}_{i'}^k), a_i^t) \right], \qquad (10)$$

where $\hat{\pi}_{i'}^k$ denotes the k-th policy embedding of agent i', which could be the teammate's policy embeddings in the cooperative environments and also could be the opponent's policy embeddings (if available) in the competitive environments. In practice, we can obtain a trajectory k by randomly selecting or employing other sampling methods from the episode trajectory space E_i of agent i, and then acquiring the agent policy embedding by the agent representation module.

In our experiments, we optimized the policy of the controlled agent using Proximal Policy Optimization (PPO) (Schulman et al. 2017), other reinforcement learning algorithms could also be used in its place.

Experiments

Multi-agent environments

We evaluate the performance of our method HMAR in two multi-agent environments (one cooperative, one competitive): *Overcooked* (Carroll et al. 2019), and *Pommerman* (Resnick et al. 2018). These environments impose stringent demands on the sequencing of agent actions, exhibiting a pronounced hierarchical structure in the state evolution. More experiment details and additional experiments are in the Appendix.

Cooperative. The *Overcooked* environment, illustrated in Figure 3 (left), is a simplified simulation of the popular video game *Overcooked* (Ghost Town Games 2016). In this setting, two agents act as chefs in a kitchen, collaborating to cook and serve dishes. The kitchen features only three types of objects: onions (yellow), dishes (white), and a cooking pot (dark grey). To complete a task, agents must place three onions into the pot, wait 20 timesteps for the soup to cook, transfer it into a dish (white), and then serve it (light grey), earning a shared reward of 20. The overarching goal is to deliver as many soups as possible within the time limit. Agents operate using six discrete actions: move up, down, left, or right; perform no operation (noop); and "interact," which triggers context-dependent effects based on the tile they face (e.g., placing an onion on a counter). Success in Overcooked hinges on agents' ability to model their partner's behavior and coordinate efficiently. This makes the environment well-suited for evaluating agent representation learning and collaborative policy execution.

Competitive. The Pommerman environment draws its inspiration from the classic console game Bomberman (W. 1983). In this experiment, we utilize the simulator configured for two agents whose initial positions are randomized close to any of the 4 corners of the board, as depicted in Figure 3 (right). At each time step, each agent has the option to choose from six possible actions: movement in any of the four directions, staying in place, or placing a bomb. The environment consists of cells that can be passages, rigid walls (dark brown cells), or wood (light brown cells). Maps are randomly generated, and there is always a guaranteed path between any two agents on the map. The objective of the task is to be the last agent standing, earning a reward of 1 for victory, while tie games result in an episodic reward of -1. When an agent places a bomb, it explodes after 10 time steps, producing flames that last for 2 time steps. These flames have the ability to destroy wood and kill agents within their blast radius. The destruction of wood can reveal either a passage or a power-up (yellow circles). Power-ups fall into three categories: those that increase the blast radius of bombs, those that increase the number of bombs an agent can place, and those that grant the ability to kick bombs. An episode of two-player *Pommerman* is finished when an agent dies or reaches 800 timesteps.

Baseline Methods

Local Information Agent Modelling (LIAM) (Papoudakis, Christianos, and Albrecht 2021): This baseline presents an encoder-decoder agent modeling method capable of extracting concise yet informative representations of modeled agents, relying solely on the local information available to the controlled agent (including its local state observations and past actions). We include LIAM for two reasons: it employs a recurrent encoder for policy representation learning and leverages local state observations in a manner akin to HMAR. Through the results, we can compare the performance of memory modeling based on recurrent encoders with tree-like hierarchical modeling in the Poincaré ball.

Alternate Modeling Framework (AMF) (Grover et al. 2018): The baseline introduces an approach through imitation learning, employing a supervised training methodology that maps observations directly to actions, effectively capturing point-based policy representations. In contrast to HMAR, which adeptly learns policy representations within the hyperbolic geometry of the Poincaré ball, the Alternate Modeling Framework (AMF) is confined to Euclidean space operations. This juxtaposition allows us to scrutinize the performance benefits of Poincaré ball-embedded policy representations as facilitated by HMAR. To ensure a fair comparison, all baselines, including our proposed HMAR framework, undergo training utilizing the Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017), enabling an equitable evaluation of their respective efficacy.

Contrastive Agent Representation Learning (CARL): The baseline inspired by (Papoudakis, Christianos, and Albrecht 2021) is a non-reconstruction baseline based on contrastive learning (Oord, Li, and Vinyals 2018). CARL utilizes the trajectories of modeled agents during training, but restricts execution to solely the trajectories of the controlled agent. Further implementation details for this baseline can be found in the Appendix. We incorporated this baseline due to its non-reconstructive approach that leverages the concept

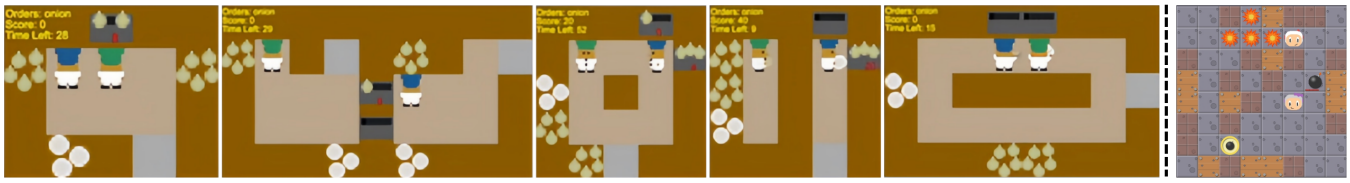

Figure 3: (Left figure) *Overcooked* environment layouts (left to right): *Cramped Room* restricts agents to a narrow space, heightening the risk of collisions. *Asymmetric Advantages* evaluates whether agents can formulate high-level strategies that leverage their individual strengths. In *Coordination Ring*, agents must synchronize their movements to navigate between the bottom-left and top-right corners of the kitchen. *Forced Coordination* necessitates the development of a joint strategy, as neither agent can complete the dish-serving task independently. *Counter Circuit* requires a non-trivial coordination pattern, where onions must be passed over the counter to the pot rather than carried around. Each layout features one or more onion and dish dispensers, offering an unlimited supply of each. (Right figure) *Pommerman* environment with 8×8 board.

of contrast and utilizes trajectory-based learning for crafting agent policy representations.

Experiment results

Cooperative. Figure 4 shows the mean episode rewards of all methods during training in five *Overcooked* environment layouts. These layouts necessitate specific sequences of actions from the agents and display a state evolution resembling a hierarchical tree-like structure. Our experimental results also validate the effectiveness of our HMAR method in learning agent policy embeddings within the Poincaré ball. It excels at extracting hierarchical information from agent trajectories, resulting in more effective agent policy embeddings. Furthermore, these embeddings encapsulate a richer set of information, which proves advantageous for the decision-making processes of agents.

Cramped Room: In this layout, the confined space makes agent collisions more likely. If agents can perceive their teammate's behavioral patterns, they can adapt their strategies accordingly to minimize interference while completing the task. As shown in Figure 4(a), HMAR-PPO typically leads to one agent learning to perform the task efficiently within the first 100 training iterations, while the other agent remains inactive, thereby avoiding early-stage collisions. The HMAR embeddings rapidly capture the teammate's policy characteristics, enabling both agents to operate independently and without interference, which results in high initial performance. As training progresses, the previously inactive agent also begins to participate, increasing the risk of collisions in the limited space and reducing task efficiency. By encoding behavioral features, such as action sequences, HMAR embeddings promote faster and more effective coordination. In contrast, although other baselines demonstrate more stable learning, significant agent interference occurs from the start.

Asymmetric Advantages: In this layout, the two agents must learn to effectively coordinate the use of both pots. If both agents rely on a single pot, task execution becomes inefficient. Therefore, the learned policy embeddings must capture rich hierarchical action information—such as the order in which pots are selected, the frequency of usage, and time allocation strategies. As shown in Figure 4(b), HMAR-PPO, which learns policy embeddings in the Poincaré ball, achieves superior performance.

Coordination Ring: The training process in this layout is similar to that in the *Cramped Room*, as depicted in Figure 4(c). Our HMAR-PPO method learns policy embeddings that extract more information about action hierarchy and action sequence relationships, enabling the agents to quickly learn to avoid collisions and efficiently complete tasks.

Forced Coordination: In this layout, the right agent manages two pots and handles soup delivery, while the left agent is tasked with supplying onions and plates. The core challenge lies in coordinating the execution sequence of actions between the two agents. Due to the limited space at the central interaction counter (only three positions), the left agent must provide ingredients in accordance with the right agent's delivery progress, while the right agent must adapt its cooking and delivery order based on the supply sequence from the left agent. As shown in Figure 4(d), HMAR-PPO facilitates faster cooperative learning between agents. Interestingly, the agents even exhibit a form of "laziness" during training. Initially, the left agent places several onions and plates on the counter, and the right agent retrieves them one by one, causing both agents to frequently shuttle through their respective corridors. Over time, however, the left agent learns to place one item at a time in the grid adjacent to the pot, while the right agent learns to pick up ingredients only from the nearest grid. Since the embeddings effectively capture the hierarchical characteristics of the behavior of the agent on the right side, and the agent on the left side has learned to "slack off" by utilizing the information provided by the policy embeddings.

Counter Circuit: In this layout, the most efficient cooperative agent policy involves passing onions through the central counter instead of both agents continuously circling the ring. As illustrated in Figure 4(e), the two agents with HMAR policy embeddings learn each other's action hierarchy characteristics, significantly improving cooperation efficiency.

Competitive. In *Pommerman* experiment, agents move to positions that could potentially eliminate their opponents by placing bombs and then quickly retreating. This results in repeated action sequences that exhibit clear hierarchical action patterns, and the evolution of states also demonstrates a tree-like hierarchical structure. To test the effectiveness of agent embeddings, we combine the opponent's embeddings with the current state as the input for the agent. If the learned opponent's agent embeddings are effective, the agent, when

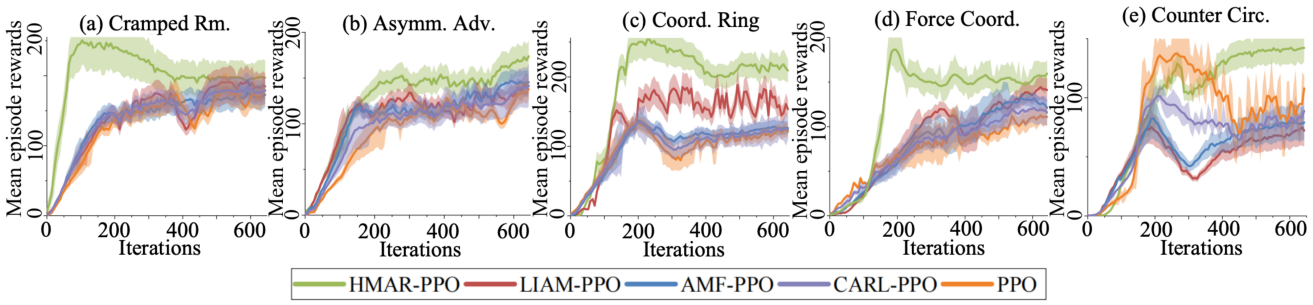

Figure 4: Average episode rewards on each layout of the *Overcooked* environment during training, shaded regions indicate the standard deviation over five training seeds.

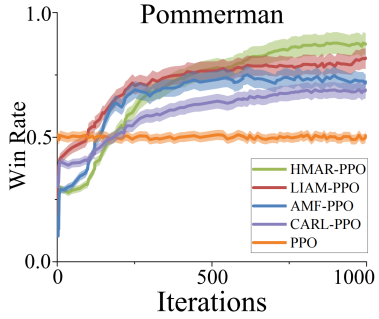

Figure 5: Average win rates of baseline agents against naive PPO agents across five training seeds in *Pommerman*.

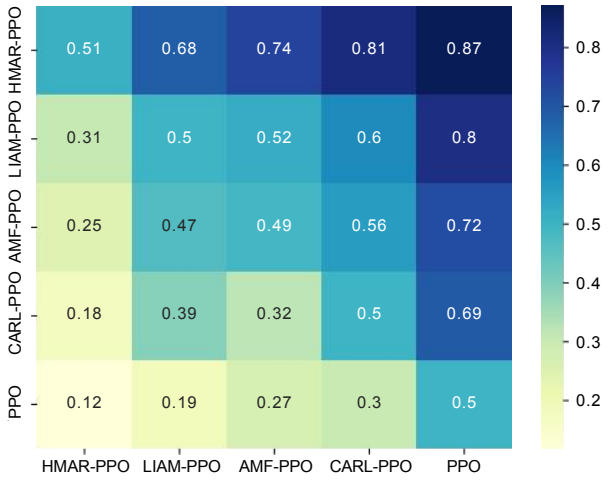

Figure 6: Win rates of baseline agents in adversarial matches after 1000 iterations in *Pommerman* environment.

making action selections, will take into account the characteristics of the opponent's policy. If the opponent's policy is ineffective, the current agent cannot obtain information about the opponent, which will directly reflect in the win rate results. We conducted two sets of experiments: (1) comparing the win rates of all baselines combined with PPO against the naive PPO algorithm and (2) comparing the win rates between all baselines (including naive PPO) after training for 1000 iterations, and all baselines are combined with PPO. In terms of winning rates, the combination of PPO with agent embeddings learned by HMAR achieved the highest winning rates in both sets of experiments. This demonstrates that HMAR is more effective at capturing hierarchical information.

The average win rates of baseline agents against naive PPO agents across five training seeds are shown in Figure 5. In the initial stages of the training process, policies are busy exploring the environment and have not yet exhibited hierarchical state evolution characteristics. Therefore, there is not enough information provided for HMAR to learn hierarchical characteristics. As the training process progresses, all baseline methods effectively learn the characteristics of the opponent's policy, with HMAR method showing superior performance. Moreover, agents in adversarial matches tend to exhibit certain patterns, meaning that policies reveal some habitual action sequences. When HMAR method captures such characteristics, it outperforms other methods when combined with PPO, resulting in higher win rates. Figure 6 presents the win rates of all baselines, including naive PPO, after 1000 training iterations. A comparison reveals that HMAR achieves over 50% win rates against all other methods. In competitive environments, where action sequences exhibit strong hierarchical structures and state transitions resemble tree-like growth, learning policy embeddings in the Poincaré ball effectively captures these hierarchical features, contributing to HMAR's advantage.

Conclusion

We rethink model agent behaviors from a geometric structure perspective in the context of MARL. Recognizing the limitations of Euclidean spaces in capturing the hierarchical and nested tree-like interdependencies among agent decisions, we propose the HMAR method. By projecting agent behaviors into a Poincaré ball and utilizing hyperbolic neural networks, the agent representations effectively preserve the structural information of the agent decision-making process. HMAR also designed a contrastive loss function to ensure clear and distinct agent representations. Experimental results demonstrated the empirical effectiveness of HMAR in both cooperative and competitive multi-agent environments. The ability of HMAR to maintain the structural integrity of agent behaviors facilitated more effective decisions.

Acknowledgments

This work was supported by the National Natural Science Foundation of China, Grant Numbers: 62476109, 62206108. This work was supported by the National Research Foundation, Singapore, under its National Large Language Models Funding Initiative (AISG Award No. AISG-NMLP-2024-004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

Albrecht, S. V.; and Stone, P. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258: 66–95.

Beltrami, E. 1868a. Teoria fondamentale degli spazii di curvatura costante. *Annali di Matematica Pura ed Applicata (1867-1897)*, 2: 232–255.

Beltrami, E. 1868b. *Teoria fondamentale degli spazii di curvatura costante memoria*. F. Zanetti.

Bläsius, T.; Friedrich, T.; Krohmer, A.; and Laue, S. 2018. Efficient embedding of scale-free graphs in the hyperbolic plane. *IEEE/ACM transactions on Networking*, 26(2): 920–933.

Bonnabel, S. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9): 2217–2229.

Cannon, J. W.; Floyd, W. J.; Kenyon, R.; Parry, W. R.; et al. 1997. Hyperbolic geometry. *Flavors of geometry*, 31(59-115): 2.

Carroll, M.; Shah, R.; Ho, M. K.; Griffiths, T.; Seshia, S.; Abbeel, P.; and Dragan, A. 2019. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32.

Cetin, E.; Chamberlain, B. P.; Bronstein, M. M.; and Hunt, J. J. 2024. Hyperbolic Deep Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

Chami, I.; Ying, Z.; Ré, C.; and Leskovec, J. 2019. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32.

Cvetkovski, A.; and Crovella, M. 2009. Hyperbolic embedding and routing for dynamic graphs. In *IEEE INFOCOM 2009*, 1647–1655. IEEE.

Dasari, S.; Ebert, F.; Tian, S.; Nair, S.; Bucher, B.; Schmeckpeper, K.; Singh, S.; Levine, S.; and Finn, C. 2020. RoboNet: Large-Scale Multi-Robot Learning. In *Conference on Robot Learning*, 885–897. PMLR.

Dhingra, B.; Shallue, C.; Norouzi, M.; Dai, A.; and Dahl, G. 2018. Embedding Text in Hyperbolic Spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, 59–69.

Dong, H.; Zhang, J.; and Zhang, C. 2023. Leveraging Hyperbolic Embeddings for Coarse-to-Fine Robot Design. In *The Twelfth International Conference on Learning Representations*.

Fang, Y.; Tang, Z.; Ren, K.; Liu, W.; Zhao, L.; Bian, J.; Li, D.; Zhang, W.; Yu, Y.; and Liu, T.-Y. 2023. Learning multi-agent intention-aware communication for optimal multi-order execution in finance. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4003–4012.

Ganea, O.; Bécigneul, G.; and Hofmann, T. 2018. Hyperbolic neural networks. *Advances in neural information processing systems*, 31.

Ghosh, D.; and Bellemare, M. G. 2020. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning*, 3556–3565. PMLR.

Ghost Town Games. 2016. Overcooked. https://store.steampowered.com/app/448510/Overcooked/.

Grover, A.; Al-Shedivat, M.; Gupta, J.; Burda, Y.; and Edwards, H. 2018. Learning policy representations in multiagent systems. In *International conference on machine learning*, 1802–1811. PMLR.

He, H.; Boyd-Graber, J.; Kwok, K.; and Daumé III, H. 2016. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, 1804–1813. PMLR.

Kleinberg, R. 2007. Geographic routing using hyperbolic space. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, 1902–1909. IEEE.

Krioukov, D.; Papadopoulos, F.; Kitsak, M.; Vahdat, A.; and Boguná, M. 2010. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3): 036106.

Krioukov, D.; Papadopoulos, F.; Vahdat, A.; and Boguná, M. 2009. Curvature and temperature of complex networks. *Physical Review E*, 80(3): 035101.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, 157–163. Elsevier.

Luo, R.; Ni, W.; and Tian, H. 2022. Visualizing multi-agent reinforcement learning for robotic communication in industrial iot networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 1–2. IEEE.

Natan, O.; and Miura, J. 2022. End-to-end autonomous driving with semantic depth cloud mapping and multi-agent. *IEEE Transactions on Intelligent Vehicles*, 8(1): 557–571.

Nickel, M.; and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30.

Nickel, M.; and Kiela, D. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International conference on machine learning*, 3779–3788. PMLR.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Papoudakis, G.; Christianos, F.; and Albrecht, S. 2021. Agent modelling under partial observability for deep reinforcement

learning. *Advances in Neural Information Processing Systems*, 34: 19210–19222.

Peng, W.; Varanka, T.; Mostafa, A.; Shi, H.; and Zhao, G. 2021. Hyperbolic deep neural networks: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, 44(12): 10023–10044.

Qu, B.; Cao, X.; Chang, Y.; Tsang, I. W.; and Ong, Y.-S. 2024. Diversifying Policies With Non-Markov Dispersion to Expand the Solution Space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Rabinowitz, N.; Perbet, F.; Song, F.; Zhang, C.; Eslami, S. A.; and Botvinick, M. 2018. Machine theory of mind. In *International conference on machine learning*, 4218–4227. PMLR.

Raileanu, R.; Denton, E.; Szlam, A.; and Fergus, R. 2018. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, 4257–4266. PMLR.

Resnick, C.; Eldridge, W.; Ha, D.; Britz, D.; Foerster, J.; Togelius, J.; Cho, K.; and Bruna, J. 2018. PommerMan: A multi-agent playground. In *CEUR Workshop Proceedings*, volume 2282. CEUR-WS.

Rusu, A. A.; Rao, D.; Sygnowski, J.; Vinyals, O.; Pascanu, R.; Osindero, S.; and Hadsell, R. 2018. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*.

Sala, F.; De Sa, C.; Gu, A.; and Ré, C. 2018. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, 4460–4469. PMLR.

Sarkar, R. 2011. Low distortion delaunay embedding of trees in hyperbolic plane. In *International symposium on graph drawing*, 355–366. Springer.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shavitt, Y.; and Tankel, T. 2008. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Transactions on Networking*, 16(1): 25–36.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Sun, S.; Qin, M.; Zhang, W.; Xia, H.; Zong, C.; Ying, J.; Xie, Y.; Zhao, L.; Wang, X.; and An, B. 2024. TradeMaster: A Holistic Quantitative Trading Platform Empowered by Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36.

Tacchetti, A.; Song, H. F.; Mediano, P. A.; Zambaldi, V.; Rabinowitz, N. C.; Graepel, T.; Botvinick, M.; and Battaglia, P. W. 2018. Relational forward models for multi-agent learning. *arXiv preprint arXiv:1809.11044*.

Tifrea, A.; Bécigneul, G.; and Ganea, O.-E. 2018. Poincaré glove: Hyperbolic word embeddings. *arXiv preprint arXiv:1810.06546*.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

W. 1983. Bomberman. https://en.wikipedia.org/wiki/Bomberman.

Walter, J. A. 2004. H-MDS: a new approach for interactive visualization with multidimensional scaling in the hyperbolic space. *Information systems*, 29(4): 273–292.

Yu, T.; and De Sa, C. M. 2019. Numerically accurate hyperbolic embeddings using tiling-based models. *Advances in Neural Information Processing Systems*, 32.

Zhang, H.; Zhang, W.; Qu, H.; and Liu, J. 2025. Enhancing human-centered dynamic scene understanding via multiple llms collaborated reasoning. *Visual Intelligence*, 3(1): 3.

Zhou, M.; Luo, J.; Villella, J.; Yang, Y.; Rusu, D.; Miao, J.; Zhang, W.; Alban, M.; Fadakar, I.; Chen, Z.; et al. 2021. Smarts: An open-source scalable multi-agent rl training school for autonomous driving. In *Conference on Robot Learning*, 264–285. PMLR.

Zintgraf, L.; Devlin, S.; Ciosek, K.; Whiteson, S.; and Hofmann, K. 2021. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*.