UNIVERSITY OF CALIFORNIA

Los Angeles

Learning Theory of Mind for Multi-agent Planning

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Qingyi Zhao

2020

ABSTRACT OF THE THESIS

Learning Theory of Mind for Multi-agent Planning

by

Qingyi Zhao
Master of Science in Computer Science
University of California, Los Angeles, 2020
Professor Song-Chun Zhu, Chair

Theory of mind (ToM) refers to the ability to understand oneself's and others' mental states. The major difficulty of incorporating ToM into multi-agent planning is updating an agent's beliefs of others' beliefs over time, which are probability distributions over distributions. In this work, we propose a novel way to model this nested belief update with a higher computational efficiency, simultaneously providing an approach to learn other agents' models. We model the belief update by a Markov probability transition, which linearly updates beliefs as distributions. This transition kernel characterizes another agent's belief update and it is learnable, thus we learn other agents' models by learning their kernels. We demonstrate the effectiveness of our algorithm in a police-thief game, showing that our agent is able to learn other agents' belief updates and intentionally change their beliefs to achieve its own goal.

The thesis of Qingyi Zhao is approved.

University of California, Los Angeles

2020

To my family and friends

who always supported me

Table of Contents

List of Figures

Acknowledgements

First of all, I would like to thank my advisor Song-Chun Zhu. It has been a great honor to work with and extremely lucky for me to be advised by Song-Chun. His guidance along the way has helped me grow both academically and as a person. I would also like to express my gratitude towards to Siyuan Qi, who has been my mentor throughout my research life. Through numerous discussions I had with Siyuan, I got to know how to be a true researcher and exposed to a great number of intriguing ideas. Next, I want to thank Professor Tao Gao, and other members in VCLA, Yixin Zhu, Baoxiong Jia, Feng Gao, Luyao Yuan, Feng Shi, Victor Zhang for their help and insightful discussions we have had.

I would like to express my sincerest gratitude to my parents, grand parents, and other family members who have supported me and given me confidence during the hard times of my life.

This work is mainly based on a draft submission to ICML 2019, with Siyuan Qi, Yingnian Wu, and Song-Chun Zhu listed as co-authors. Siyuan Qi led the project and contributed the most to problem formulation, algorithm design, experiments, and manuscript writing. Professor Yingnian Wu and Song-Chun Zhu offered directional suggestions and guidance along the way.

CHAPTER 1

Introduction

Multi-agent systems, ranging from two-player games to the human society, have been studied across many domains. For individual agents to maximize their values in such environments, they must learn to interact with and against others, as well as understand the consequences of their actions.

Most multi-agent planning approaches focus on modeling other agents' policies based on physical world states [33, 15, 9, 8]. However, besides the physical states, humans also reason about other agents' mental states including their beliefs. This ability to understand oneself's and others' mental states and reason about their behaviors, is commonly referred to as the theory of mind (ToM).

We argue that there are two distinctions between ToM and non-ToM agents. 1) ToM agents make predictions in a higher dimension (both physical and mental states). 2) ToM agents intentionally change other agents' mental states, *i.e.*, beliefs about the world. In this way, a ToM agent *changes* others' future behaviors to maximize its own value. In contrast, traditional approaches only *react to* predicted future behaviors of other agents. A representative line of work to incorporate ToM into planning algorithms is interactive partially observable Markov Decision processes (I-POMDPs) [10].

Figure 1.1 shows a ToM planning framework. In this framework, a ToM agent keeps beliefs (probability distributions) about the world state by Bayesian updates over time: at every time step, it updates the prior belief after performing an action and receiving an observation. It also keeps high-level beliefs about other agents' beliefs, which are represented by distributions of distributions. They are computed by a *nested Bayesian update*, which

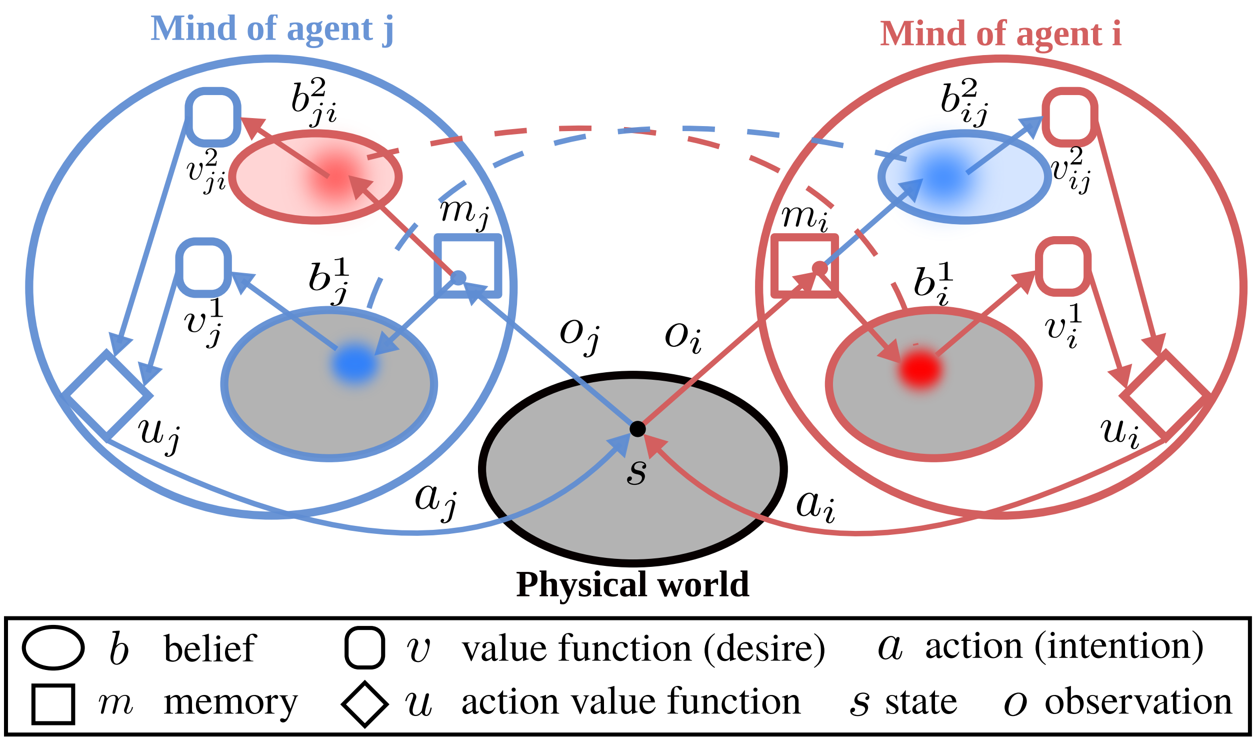


Figure 1.1: A ToM agent observes the world s and save the observation o into its memory m. The memory includes $o_{1:t}$ its past observations, and $a_{1:t-1}$ the performed actions. Based on the memory, it updates is belief b^1 about the world, belief b^2 about other agents' beliefs. Each level l of belief b^l has a corresponding value function $v^l(\cdot)$, and value functions are combined into an action value function u. Finally, the agent chooses an optimal action a based on u and changes the world state.

involves Bayesian updates of other agents' lower-level beliefs.

However, exact inference for nested belief updates is computationally very difficult for several reasons. These reasons include 1) other agents' model (*e.g.*, observation function) are required to compute the nested belief updates. 2) Belief updates for world states suffer from the curse of dimensionality, and nested belief updates aggravate this issue. In general, only approximate belief updates are computable.

To avoid the first problem, most methods assume that other agents' models are known [4, 6, 5, 7, 25, 20]. A more recent work [12] removes this constraint by adding a prior distribution to other agents' models.

To alleviate the second problem, various approximation methods for the belief update are proposed, including particle filter [5], state space pruning [6], and nested MDPs [13].

The major contribution of this work is a novel way to approximate the belief update, at the same time providing an alternative approach to learn other agents' models. By decomposing the belief update, we identify and approximate a "belief dynamics" term that is particularly computationally costly. The belief dynamics predicts how other agents' beliefs will change after our agent performs an action. We model this process by a Markov probability transition. The transition kernel linearly transforms the current belief to a predicted belief, thus greatly reducing the computational complexity. We can show that there always exists a kernel that transforms the true beliefs, hence theoretically this approximation can be exact. The kernel is also learnable, *e.g.*, by generator neural networks. Since this kernel characterizes another agent's belief update, our agent essentially learns the other agents' models by learning their kernels.

We demonstrate the effectiveness of our algorithm in a police-thief game, in which the police agent needs to catch the thief while hiding its own identity from the thief. In the experiments, we show that 1) our agent can accurately estimate the beliefs of other agents. 2) Our agent can learn meaningful values over physical states and mental states. 3) Most importantly, our agent is able to intentionally change other agents' beliefs to achieve high values, outperforming other state-of-the-art multi-agent planning algorithms.

CHAPTER 2

Literature Review

Theory of mind has first been studied in psychology and cognitive science [31]. It has been shown that the ability to perform mental simulations of others increases rapidly since the young-infant phase [32, 11, 16, 19]. This allows reasoning about others' mind, and it is vital in a multi-agent environment because each agent's choice affects the payoff of other agents [30, 3].

This stimulates a growing interest in the AI community to build algorithms that exhibit this ability [2, 14]. For example, Bayesian Theory of Mind (BToM) [2, 1] predicts the mental states of humans. [21] proposed ToMnet, a neural network to predict the characteristic of an observed agent and its future behaviors. These are perception models that do not involve planning.

The most representative series of work on ToM planning is I-POMDP [4, 6, 5, 7, 25, 20], which extends the traditional POMDPs. It augments world states to interactive states to include beliefs of the intentional model of other agents (belief, reward function, observation function, etc.). However, solving I-POMDP can be extremely expensive and inefficient. The generalization to interactive states greatly increases the dimension of the state space, and this curse of dimensionality is exacerbated by the nested belief reasoning among agents.

Methods have been proposed to approximate I-POMDP. For example, I-PF [5] approximates the belief updates by particle filters. I-PBVI [6] constrains the interactive state space by computing a finite set of beliefs reachable from the initial belief over a certain horizon. I-POMDP Lite [13] uses a nested MDP to model other agents to approximate the exact I-POMDP policy.

The above approximation methods all assume that the agent knows exactly the models, except beliefs, of other agents. A more recent work [12] removes this constraint. It performs belief update using Bayesian inference and particle filtering by sampling other agents' models from prior distributions.

In our work, the agent approximates the belief update by a Markov transition kernel, which is different from all prior methods. This kernel acts linearly on beliefs, hence greatly reducing the computational complexity. This transition kernel is learnable, hence our agent can learn the models of other agents without specified prior knowledge. This also provides a different perspective from prior methods to learn other agents' models.

CHAPTER 3

Algorithm

3.1 Theory-of-mind Belief Update

To act in uncertain environments due to noisy/partial observation, an agent tracks the physical world state s over time based on the actions it performed and its past observations. At each time step t, an agent i keeps a *belief* $b_{i,t}$, which is a probability distribution of the world state s_t given its memory $m_{i,t}$. The memory includes its past observations, $o_{i,1:t}$, and the performed actions, $a_{i,1:t-1}$.

In the rest of the thesis, we will use numbered superscripts to indicate the ToM level of the variables (*e.g.*, first level beliefs b^1). A first-order ToM agent tracks not only the physical world state, but also the mental states of other agents. Specifically, it tracks its state $s = (s^0, b^1)$ over time, where b^1 is all agents' first-level beliefs of the world state s^0. In other words, a first-order ToM agent i maintains two types of beliefs: first- and second-level beliefs. They are probability distributions of s^0 and b^1, respectively. The first-level belief is formulated as the probability of the world state given its past observation and actions:

$$b^1_{i,t}(s^0_t) = p(s^0_t | o_{i,1:t}, a_{i,1:t-1}) \tag{3.1}$$

The second-level belief $b^2_{ij,t}$ is defined as agent i's belief about agent j's first-level belief $b^1_{j,t}$:

$$b^2_{ij,t}(b^1_{j,t}) = p(b^1_{j,t} | o_{i,1:t}, a_{i,1:t-1}), \text{for } j \neq i. \tag{3.2}$$

At every time step t, an agent i updates its belief $b_{i,t-1}$ to $b_{i,t}$, according to the action $a_{i,t-1}$ performed at last time step and the observation $o_{i,t}$ received afterwards. This is called belief update, which we will discuss in details in the rest of this section. In general, the agent estimates the physical and mental states by Bayes filtering, which updates its belief each time an action is performed and a new observation arrives.

3.1.1 First-level Belief Update

The first-level belief of agent i about the world state at time t is $b^1_{i,t}(s^0_t) = p(s^0_t | o_{i,1:t}, a_{i,1:t-1})$. The Bayes filtering to update the belief at time t can be decomposed into a two-step process:

• *Prediction.* The agent updates its previous belief $b^1_{i,t-1}(s^0_{t-1})$ after taking an action $a_{i,t-1}$ by predicting how the state s^0_{t-1} will change:

$$\begin{aligned} & p(s^0_t | o_{i,1:t-1}, a_{i,1:t-1}) \\ & = \int_{s^0_{t-1}} p(s^0_t, s^0_{t-1} | o_{i,1:t-1}, a_{i,1:t-1}) ds^0_{t-1} \\ & = \int_{s^0_{t-1}} p(s^0_t | s^0_{t\text{-}1}, o_{i,t\text{-}1}, a_{i,t\text{-}1}) p(s^0_{t\text{-}1} | o_{i,1:t\text{-}1}, a_{i,1:t\text{-}2}) ds^0_{t\text{-}1} \\ & = \int_{s^0_{t-1}} \underbrace{p(s^0_t | s^0_{t-1}, o_{i,t-1}, a_{i,t-1})}_{\text{world dynamics}} \underbrace{b^1_{i,t-1}(s^0_{t-1})}_{\text{previous belief}} ds^0_{t-1} \end{aligned} \tag{3.3}$$

We can see that the agent updates its previous belief $b^1_{i,t-1}(s^0_{t-1})$ by applying a stochastic state transition function $p(s^0_t | s^0_{t-1}, o_{i,t-1}, a_{i,t-1})$. At this time, the agent has not received the new observation $o_{i,t}$. This prediction step computes how first-level beliefs will change after performing an action. Note that although we have uncertainty about the true world state, but this distribution itself (*i.e.*, the belief) is deterministic for a certain time step.

• *Correction.* After receiving a new observation $o_{i,t}$, the agent corrects its prediction from the last step :

$$\begin{aligned} b^1_{i,t}(s^0_t) &= p(s^0_t | o_{i,t}, o_{i,1:t-1}, a_{i,1:t-1}) \\ &= \alpha\, p(s^0_t, o_{i,t} | o_{i,1:t-1}, a_{i,1:t-1}) \\ &= \alpha\, p(o_{i,t} | s^0_t) \underbrace{p(s^0_t | o_{i,1:t-1}, a_{i,1:t-1})}_{\text{first-level prediction}} \end{aligned} \tag{3.4}$$

where α is a normalizing constant as we apply the Bayes rule, $p(o_{i,t} | s^0_t)$ is the likelihood of observation $o_{i,t}$, and $p(s^0_t | o_{i,1:t-1}, a_{i,1:t-1})$ is the updated belief from the prediction step.

3.1.2 Second-level Belief Update

The second-level belief gets updated in a similar way. At each time step t, agent i maintains a second-level belief about agent: $b^2_{ij,t}(b^1_{j,t}) = p(b^1_{j,t}|o_{i,1:t}, a_{i,1:t-1})$, for $j \neq i$. The Bayes filtering is the same as the first-level belief update, except that the states are replaced by first-level beliefs:

- *Prediction*:

$$\begin{aligned} &p(b^1_{j,t}|o_{i,1:t-1}, a_{i,1:t-1}) \\ &= \int_{b^1_{j,t-1}} \underbrace{p(b^1_{j,t}|b^1_{j,t-1}, o_{i,t-1}, a_{i,t-1})}_{\text{belief dynamics}} b^2_{ij,t-1}(b^1_{j,t-1}) db^1_{j,t-1} \end{aligned} \tag{3.5}$$

Similar to the first-level belief update, the agent updates the previous belief $b^2_{ij,t-1}(b^1_{j,t-1})$ by applying a stochastic transition function $p(b^1_{j,t}|b^1_{j,t-1}, o_{i,t-1}, a_{i,t-1})$ defined on the first-level belief of agent j. We call this transition function the *belief dynamics*. It takes the observation as input since it contains information about the state, and the first-level belief changes differently under different states even if the same action is performed.

In principle, the belief dynamics should follow a similar Bayesian update as the first-level belief update of agent j itself. However, exact inference of the belief dynamics needs to marginalize out all possible j's observations and actions, and then perform a lower-level belief update. This is computationally very expensive or intractable. We discuss our proposed approximation method in Section 3.1.3.

- *Correction*. The belief is then updated by the observation:

$$\begin{aligned} b^2_{ij,t}(b^1_{j,t}) &= p(b^1_{j,t}|o_{i,t}, o_{i,1:t-1}, a_{i,1:t-1}) \\ &= \alpha\, p(o_{i,t}|b^1_{j,t}) \underbrace{p(b^1_{j,t}|o_{i,1:t-1}, a_{i,1:t-1})}_{\text{second-level prediction}} \end{aligned} \tag{3.6}$$

For simplicity, we still use α to represent the normalizing constant, with a value different from the first-level belief.

3.1.3 Belief Dynamics

The belief dynamics predicts how a belief of another agent, in the form of a probability distribution, will change stochastically when actions are performed over time. However, exact inference for nested belief update is computationally very expensive or intractable as discussed in Section 3.1.2. Approximated solutions have been proposed, such as particle filters [5] and bounded policy iteration [25]. Here we propose an alternative solution that is simple but effective.

The existing methods model the belief state transition in a general way similar to world state transitions (*e.g.*, particle filter). Let us denote the current belief as b_t, the predicted belief after performing an action as $\hat{b}_{t+1}$. The key observation of our method is that this Bayesian update process can always be described by a probability transition kernel. In other words, we can always find a probability transition kernel to transform b_t to $\hat{b}_{t+1}$. Formally, we have the following proposition.

Proposition 1. [1] *Let $\mathcal{S}$ be a measurable space and $s_t, s_{t+1} \in \mathcal{S}$. $\forall b_t(s_t) = p(s_t|o_{1:t}, a_{1:t-1})$ and $\hat{b}_{t+1}(s_{t+1}) = p(s_{t+1}|o_{1:t}, a_{1:t})$, there exists a Markov kernel $\kappa(s_t, s_{t+1}) \in \mathcal{S} \times \mathcal{S} \to [0,1]$ such that (1) $\forall s_{t+1}$, $\hat{b}_{t+1}(s_{t+1}) = \int_{s_t} b_t(s_t)\kappa(s_t, s_{t+1})ds_t$, and (2) $\forall s_t$, $\int_{s_{t+1}} \kappa(s_t, s_{t+1})ds_{t+1} = 1$.*

Proof. Since s_t is independent of a_t, we have

$$\begin{aligned}
\hat{b}_{t+1}(s_{t+1}) &= p(s_{t+1}|o_{1:t}, a_{1:t}) \\
&= \int_{s_t} p(s_{t+1}|s_t, o_{1:t}, a_{1:t})p(s_t|o_{1:t}, a_{1:t})ds_t \\
&= \int_{s_t} p(s_{t+1}|s_t, o_{1:t}, a_{1:t})p(s_t|o_{1:t}, a_{1:t-1})ds_t \\
&= \int_{s_t} p(s_{t+1}|s_t, o_{1:t}, a_{1:t})b_t(s_t)ds_t
\end{aligned}$$

Obviously $\forall s_t$, $\int_{s_{t+1}} p(s_{t+1}|s_t, o_{1:t}, a_{1:t}) = 1$.

Hence $\kappa(s_t, s_{t+1}) = p(s_{t+1}|s_t, o_{1:t}, a_{1:t})$ is a kernel that satisifies the conditions. □

[1] *For simplicity, we omit the agent indices in the proposition.*

This introduces a general linear form for the belief dynamics that can be extended to higher levels. Representing the beliefs by vectors, we can re-write the belief dynamics as:

$$\hat{b}^1_{j,t} = \kappa_{t-1} b^1_{j,t-1} \tag{3.7}$$

Parametrized by observations and actions, the kernel $\kappa(o, a)$ transforms the previous belief to the predicted belief. The kernel $\kappa(o, a)$ is to be learned. For example, it can be generated by a generator neural network. The kernel acts on the belief linearly, but the kernel itself is non-linear in its parameters (*i.e.*, the observations and actions). Due to its simple and general form, it is quite powerful and computationally favorable.

Since it is difficult to learn a perfect kernel, we can further add a noise term to this belief transition:

$$\hat{b}^1_{j,t} = \kappa_{t-1} b^1_{j,t-1} + \epsilon \tag{3.8}$$

Assuming the noise follows a multivariate Gaussian distribution as a convenient approximation, we have $\epsilon \sim \mathcal{N}(0, \Sigma_D)$ where Σ_D is the covariance matrix. Equivalently, we have $\hat{b}^1_{j,t} \sim \mathcal{N}(\kappa_{t-1} b^1_{j,t-1}, \Sigma_D)$ where $\kappa_{t-1} \triangleq \kappa(o_{i,t-1}, a_{i,t-1})$. After adding the noise, we normalize $b^1_{j,t}$ to ensure that it remains to be a distribution.

Given the above belief dynamics, we can efficiently compute the second-level belief update. If at the last time step we have $b^1_{j,t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$, then the prediction step (Eq. 3.5) is a convolution of two Gaussian distributions. Hence the result will still be a Gaussian distribution. According to the second-level belief prediction (Eq. 3.5) and correction (Eq. 3.6), we have the following conclusions:

- The *prediction* step gives $b^1_{j,t} \sim \mathcal{N}(\mu_p, \Sigma_p)$, where

$$\mu_p = \kappa_{t-1} \mu_{t-1} \tag{3.9}$$

$$\Sigma_p = \Sigma_D + \kappa_{t-1} \Sigma_{t-1} \kappa_{t-1} \tag{3.10}$$

Hence we can obtain the μ_p and Σ_p by applying the transition kernel κ_{t-1}. To compute the correction step, we need a observation model. We can first learn an *belief estimation model*

$\widetilde{b^1_{j,t}} = f(o_{i,t})$ that directly estimates the beliefs from observations up to a Gaussian noise: $\widetilde{b^1_{j,t}} \sim \mathcal{N}(b^1_{j,t}, \Sigma_o)$. Then we can rewrite the observation model $p(o_{i,t}|b^1_{j,t})$ as $p(\widetilde{b^1_{j,t}}|b^1_{j,t})$, which is a Gaussian distribution. Then the correction step is computing the predictive posterior given a Gaussian prior and a Gaussian likelihood. Since a multivariate Gaussian is the conjugate prior of itself, we have:

- The *correction* step gives $b^1_{j,t} \sim \mathcal{N}(\mu_t, \Sigma_t)$, where

$$\mu_t = (\Sigma_p^{-1} + \Sigma_o^{-1})^{-1}(\Sigma_p^{-1}\mu_p + \Sigma_o^{-1}\mu_o) \tag{3.11}$$

$$\Sigma_t = (\Sigma_p^{-1} + \Sigma_o^{-1})^{-1} \tag{3.12}$$

where μ_o is directly predicted by the belief estimation model. Σ_o is learned in the training phase of the belief estimation model by computing the covariance between predicted beliefs and ground truth beliefs.

The above equations give an efficient way to compute the second-level belief update. The above process agrees with the Kalman filter. When the state space is continuous, the belief is a continuous function instead of a discrete vector. Then infinite-dimensional Kalman filter can be adopted [22].

3.2 Theory-of-mind Planning

Based on the belief about the world state and other agents' mental states, a ToM agent chooses an optimal action based on a value function. This value function is defined on the belief space, and it is convex and piecewise-linear [10]. Usually this function is learned by value iteration. However, this is very hard due to the curse of dimensionality.

In the reinforcement learning literature, it is common to learn approximated value functions [26]. Here we approximate this value function by a linear function of the beliefs with intuitive semantic meanings. Specifically, we define the first-level value function as:

$$v_i^1(b_i^1) = \int_{s^0} b_i^1(s^0) v_i^0(s^0) ds^0 \tag{3.13}$$

where $v_i^0(s^0)$ is a value function defined on true world states. The second-level value function $v_{ij}^2(b_{ij}^2)$ measures agent i's value of its second-level belief b_{ij}^2 on agent j's belief. The second-level value function is similarly defined as:

$$\begin{aligned} v_{ij}^2(b_{ij}^2) &= \int_{s^0}\int_{b_j^1} b_{ij}^2(b_j^1)b_j^1(s^0)v_{ij}^0(s^0)db_j^1 ds^0 \\ &= \int_{s^0} \underbrace{\int_{b_j^1} b_{ij}^2(b_j^1)b_j^1(s^0)db_j^1}_{\text{Expectation of first-level belief}} \quad v_{ij}^0(s^0)ds^0 \\ &= \int_{s^0} E_{b_{ij}^2}[b_j^1]\, v_{ij}^0(s^0)ds^0 \end{aligned} \tag{3.14}$$

It is grounded to the actual mental states by the zero-level value function $v_{ij}^0(s^0)$, which is the value of agent i when agent j believes that the state is in s^0 with probability 1. Notice the difference between $v_i^0(s^0)$ and $v_{ij}^0(s^0)$: $v_i^0(s^0)$ is defined on the physical state while $v_{ij}^0(s^0)$ is defined on the mental state.

Then the first-order ToM agent i at time t chooses an optimal action $a_{i,t}^*$ that maximizes its future value, which combines the first- and second-level value functions:

$$\begin{aligned} a_{i,t}^* =& \operatorname*{argmax}_{a_{i,t}} u_i^1(b_{i,t}^1, a_{i,t}) + \sum_{j\neq i} u_i^2(b_{ij,t}^2, a_{i,t}) \\ =& \operatorname*{argmax}_{a_{i,t}} \underbrace{\int_{b_{i,t+1}^1} p(b_{i,t+1}^1|b_{i,t}^1, a_{i,t})v_i^1(b_{i,t+1}^1)db_{i,t+1}^1}_{\text{expected future physical state value}} \\ & \underbrace{+\sum_{j\neq i}\int_{b_{ij,t+1}^2} p(b_{ij,t+1}^2|b_{ij,t}^2, a_{i,t})v_{ij}^2(b_{ij,t+1}^2)db_{ij,t+1}^2}_{\text{expected future mental state value}} \end{aligned} \tag{3.15}$$

This way the agent considers the change of other agents' beliefs after taking an action. This is particularly important since it enables the first-order ToM agent to intentionally change other agents' beliefs. This changes others' future behaviors and thus maximizes the agent's own value.

3.3 Learning

3.3.1 Learning Belief Dynamics

To predict the future beliefs, the agent generates the transition kernel $\kappa(o_{i,t-1}, a_{i,t-1})$ given its observation $o_{i,t-1}$ and action $a_{i,t-1}$. Each column of this transition kernel κ sums to 1. This can be practically implemented as a generator neural network, which takes the observation and action as inputs and generates all the columns for κ. To ensure that each column sums to 1, a softmax activation function can be added before the final output of each column.

3.3.2 Learning Estimation Model

The belief estimation model $\widetilde{b^1_{j,t}} = f(o_{i,t})$ estimates the first-level belief of agent j given agent i's observation at time t. This can be learned by any classification model that outputs a probability for each state given the observation. The model can be trained by optimizing the difference (*e.g.*, mean squared error, cross entropy) between the ground truth and estimated belief of agent j.

3.3.3 Learning Zero-level Value Functions

Since the beliefs of agent i (*i.e.*, $b^1_{i,t}$ and $b^2_{ij,t+1}$) themselves are updated deterministically by Bayes filtering, the stochastic belief prediction $p(b^1_{i,t+1}|b^1_{i,t}, a_{i,t})$ in Eq.3.15 is given by the Dirac delta function $\delta(b^1_{i,t+1} - \hat{b}^1_{i,t+1})$, where $\hat{b}^1_{i,t+1}$ is the belief after the performing the prediction step. Hence for first-level future value function we have:

$$\begin{aligned} u^1_i(b^1_{i,t}, a_{i,t}) &= \int_{b^1_{i,t+1}} p(b^1_{i,t+1}|b^1_{i,t}, a_{i,t}) v^1_i(b^1_{i,t+1}) db^1_{i,t+1} \\ &= \int_{b^1_{i,t+1}} \delta(b^1_{i,t+1} - \hat{b}^1_{i,t+1}) v^1_i(b^1_{i,t+1}) db^1_{i,t+1} \\ &= v^1_i(\hat{b}^1_{i,t+1}(a_{i,t})) = \int_{s^0} \hat{b}^1_{i,t+1}(s^0) v^0_i(s^0) ds^0 \end{aligned} \tag{3.16}$$

where $\hat{b}^1_{i,t+1}$ is the predicted future belief by performing the prediction step in the Bayes filtering after taking action $a_{i,t}$. Similarly, for the second-level future value function:

$$\begin{aligned} u_i^2(b^2_{ij,t}, a_{i,t}) &= v^2_{ij}(\hat{b}^2_{ij,t+1}(a_{i,t})) \\ &= \int_{s^0} E_{\hat{b}^2_{ij,t+1}}[b^1_{j,t+1}]\, v^0_{ij}(s^0)ds^0 \end{aligned} \tag{3.17}$$

Finally, the combined value function can be re-written as an inner product of beliefs and zero-level value functions:

$$q_i^1(b^1_{i,t}, a_{i,t}) + \sum_{j\neq i} q_i^2(b^2_{ij,t}, a_{i,t}) = \langle \hat{\boldsymbol{b}}, \boldsymbol{v^0} \rangle \tag{3.18}$$

where $\hat{\boldsymbol{b}}$ is the concatenation of $\hat{b}^1_i(s^0)$ and $E_{\hat{b}^2_{ij,t+1}}[b^1_{j,t+1}]$ for all $j \neq i$, and $\boldsymbol{v^0}$ is the concatenation of $v^0_i(s^0)$ and $v^0_{ij}(s^0)$ for all $j \neq i$.

From reinforcement learning's perspective, this formulation is a function approximation of the action-value functions. Specifically, $\hat{\boldsymbol{b}}$ can be interpreted as the feature vector extracted from the state, and $\boldsymbol{v^0}$ is the weight. Hence learning the zero-level value functions can be achieved by applying existing algorithms [26, 28, 27, 23] to learn the weights for a linearly approximated value function.

CHAPTER 4

Experiments

We test different multi-agent planning algorithms in a police-thief game as shown in Figure 4.1 and Figure 4.2. There are three types of agents in this game: a police (the player) controlled by the tested algorithm, a thief (the game engine) controlled by a zero-order ToM policy, and several passersby that wander around randomly. The goal of the police is catching the thief by colliding with it, while the goal of the thief is to escape until the game exceeds a maximum time limit. In this game, the thief does not know which agent is the police, so it needs to maintain a belief of every agent being the police to escape. The police knows who is the thief, but it needs to hide its identity to prevent the thief from escaping. We benchmark the police's success rate on catching the thief to evaluate the multi-agent planning methods.

4.1 Environment

We adopt the Multi-Agent Particle Environment [18], which is a two-dimensional world with continuous space and discrete time. We use a closed-world setting: an agent will reappear on the opposite side of the world after crossing a boundary. The environment is physically simulated: an agent can accelerate, and agents can collide with each other. The observation for an agent is the positions of other agents, and the allowed actions are the accelerations in four directions. The police receives a 1.0 reward when it collides with the thief, otherwise a -0.1 reward at each time step.

4.2 Thief's Belief Update

As mentioned above, we use a zero-order ToM thief to recognize potential police and escape. The thief maintains a first-level belief about each agent's identity. In the beginning, the belief for each agent is uniform (0.5 probability being police for every agent). At each time step, the belief is updated by Bayes filtering given by Eq. 3.3 and 3.4. Since an agent's identity cannot be changed by the thief's actions, the prediction step has no effect on the belief. For the correction step, we compute the observation likelihood as $p(o_{i,t}|s_t^0) = p(\theta|\mu, k)$. Here θ is the relative angle between the velocity of an agent and the line connecting the agent and the thief. $p(\theta|\mu, k) = \frac{e^{kcos(\theta-\mu)}}{2\pi I_0(k)}$ is the von Mises distribution. At every time step, the thief runs away from the agent that has the highest probability being the police.

We designed two settings in our experiments: slow-thief and fast-thief. In the slow thief setting, the thief will have a smaller acceleration than that of the police, and a directly chasing policy is sufficient to catch the thief. In the fast thief setting, the thief can easily escape from a direct chaser.

4.3 Comparative Methods

We compare our method (full/ablated) with an existing ToM method and a state-of-the-art multi-agent reinforcement learning algorithm. We use the following algorithms to learn the police's policy for benchmarking:

• ToM-gt. This is our ToM planning agent given the ground truth belief of the thief (computed by Bayesian filtering) at every time step. It does not perform belief updates, but it needs to learn the zero-level value functions. This serves as an ablative version of our full model.

• ToM. This is our full model. It learns the zero-level value functions, belief dynamics, and belief estimation models. It performs belief updates by Bayesian filtering. To learn the zero-level value functions, we use true online TD(λ) [23]. Therefore the value functions are estimated and updated at every time step during each episode. We use a 3-layer fully connected neural network for both the transition kernel generator and the belief estimation

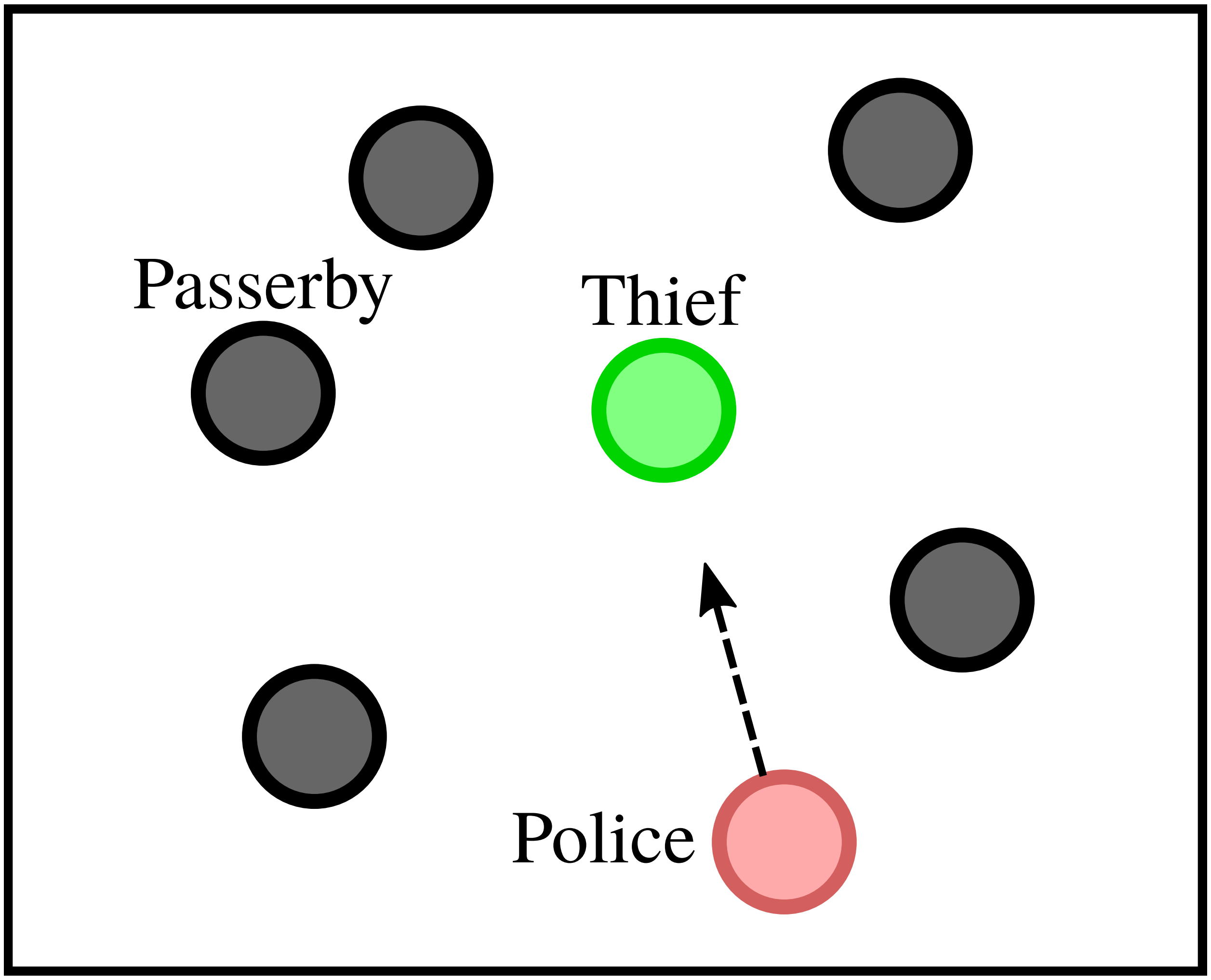


Figure 4.1: The ground truth game setting. It shows the ground truth identities of all agents

model. The size of hidden layers is 64 for the kernel generator, and 32 for belief estimation.

• MADDPG [15]. This is a state-of-the-art multi-agent reinforcement learning algorithm that extends deep deterministic policy gradient (DDPG) [24] to a multi-agent setting.

• MToM [29]. Another theory-of-mind planning approach. However, this approach only models the belief of policies rather than the belief of world states. We use the deep Q-Network (DQN) [17] for the initial policy estimation of MToM.

• Direct chasing. This agent always chooses the action that will minimize the distance between itself and the thief.

• Passerby. The same random walker as the other passersby.

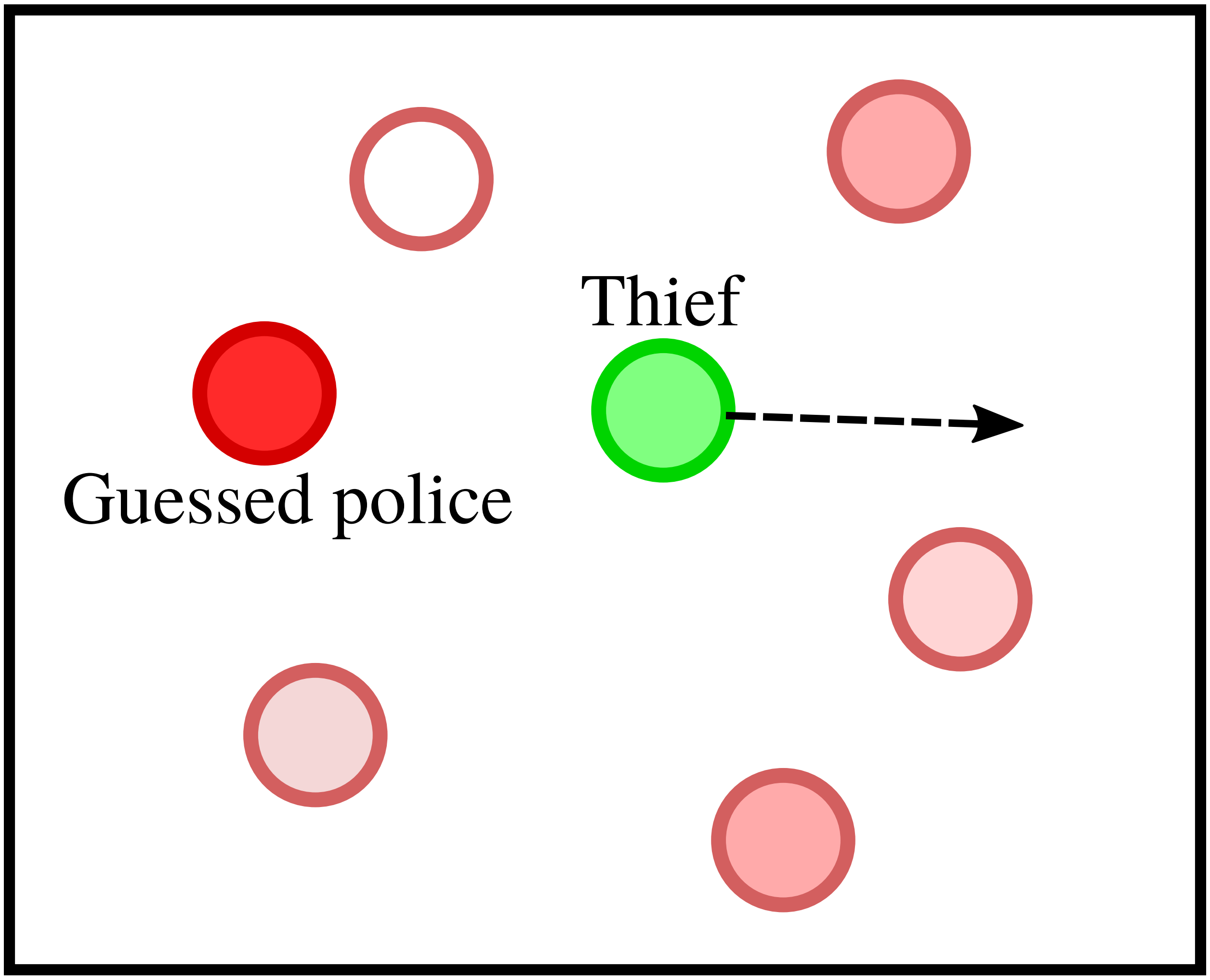
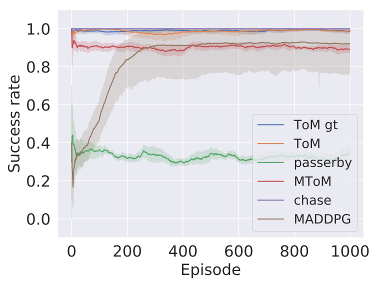


Figure 4.2: The belief game setting. It shows the thief's belief of each agent being the police (player). Darker colors indicates higher probabilities.

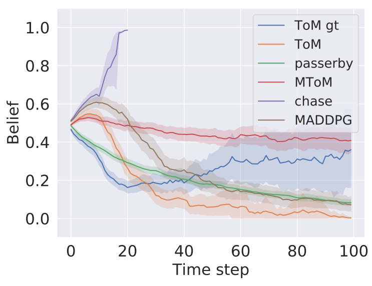
For simplicity, we only use the positions of the thief and the police itself as observations for every method. In our method, we also compute the distance between these two positions as a feature. The maximum episode length is 100, and each algorithm is trained for 1000 episodes.

4.4 Experiment Results

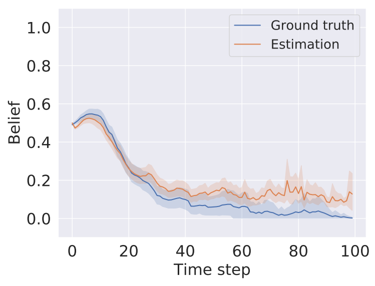
We show our quantitative experiment results in Figure 4.3 and Figure 4.4. In both settings, our method achieves a high success rate and outperforms other methods.



(a) Average success rate



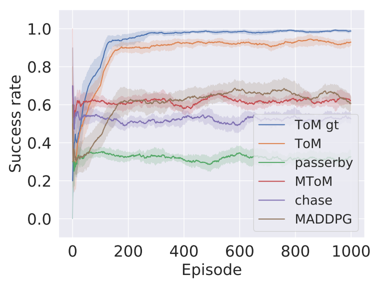
(b) Thief's belief



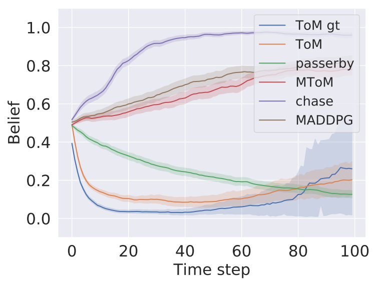
(c) Estimated belief

Figure 4.3: Experiment results in the **slow**-thief setting. (a) shows the average success rate of catching the thief in the training stage. (b) shows the probability within an episode that the thief predicts the agent is the police. The curves are averaged over 20 final episodes during training. From the beliefs we can see that the strategies of different methods vary, but they can achieve similar performance (as shown in (b)) in the slow-thief setting. (c) shows the estimated belief of the thief by our full ToM model within an episode, averaged over the final 20 episodes during training. The estimated beliefs are obtained by the two-level belief update, which employs a learned belief dynamics and belief estimation model. From the figure we can see that our agent is able to estimate the thief's belief within a small error. All the plots use data generated from 10 independent training trials.

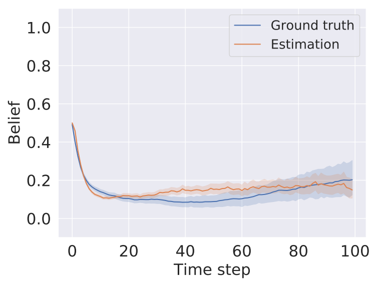
Slow-thief setting In the slow-thief setting, different methods show different belief curves (Figure 4.3b) but they achieve similar performance (Figure 4.3a). This is because the thief is easy to catch in this setting, so different methods converge to different types of policies. The value functions for ToM converges very quickly (within an episode) since we use true online TD(λ). It updates the value functions at every time step during every episode. In the beginning, the belief dynamics and estimation model are inaccurate, but it does not affect the performance. Even if the thief correctly recognizes the police, it can be caught by direct chasing. Typically, the thief is caught within 20 steps for all methods, and we can see a belief raise in that phase. In some episodes, the agent fails to catch the thief due to some random behavior. Therefore the curves go down after 20 steps similarly to the curve of the passerby (a random walker).



(a) Average success rate



(b) Thief's belief



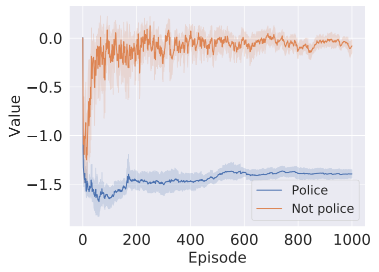
(c) Estimated belief

Figure 4.4: Experiment results in the **fast**-thief setting. (a) shows that our ToM agent significantly outperforms other methods, achieving a success rate close to 1. (b) shows that our agent intentionally lowers the thief's belief thus it achieves the high success rate. The other methods learn to chase the thief, hence they are recognized by the thief (the belief goes higher as time evolves). (c) shows that our agent is able to estimate the thief's belief within a small error.

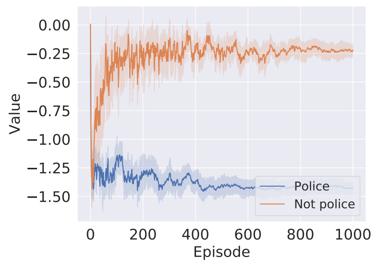
Fast-thief setting In the fast-thief setting as a contrast, the police needs to hide its identity to catch the thief with a high success rate. From Figure 4.4b we can see that the comparative non-ToM methods all converge to a greedy chasing behavior in this setting. The thief recognizes the police and escapes. The success rate is around 0.6 for non-ToM methods (including direct chasing), which is slightly higher than random walk that has a success rate of 0.4.

On the other hand, our method achieves a success rate close to 1 by hiding its identity. From the learning curve (Figure 4.4a) we can see that it takes some time for our ToM agent to learn the belief dynamics and estimation model to achieve the final performance. Our ToM agent can intentionally lower the belief of the thief to achieve its goal. Comparing with the passerby (a random walker) we can see that the belief curve of our agent has a sharp decline in the beginning. Hence this lowering is not a consequence of random movements. As a final result, our ToM agent significantly outperforms all other methods.

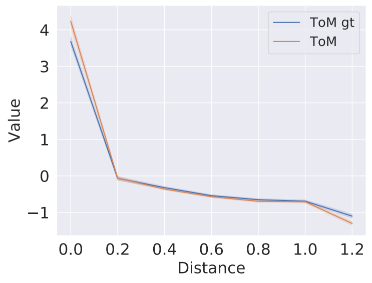
In both settings, the second-level belief update estimates the belief of the thief quite accurately as shown in Figure 4.3c and 4.4c. Qualitative results are shown in Figure 4.6, and



(a) Values of belief in ToM-gt training



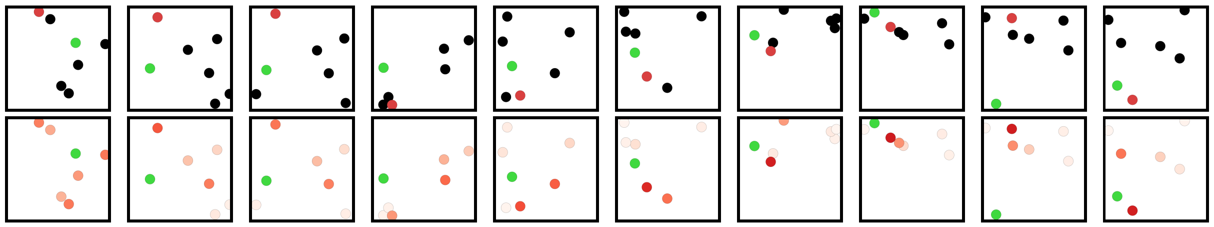
(b) Values of belief in ToM training



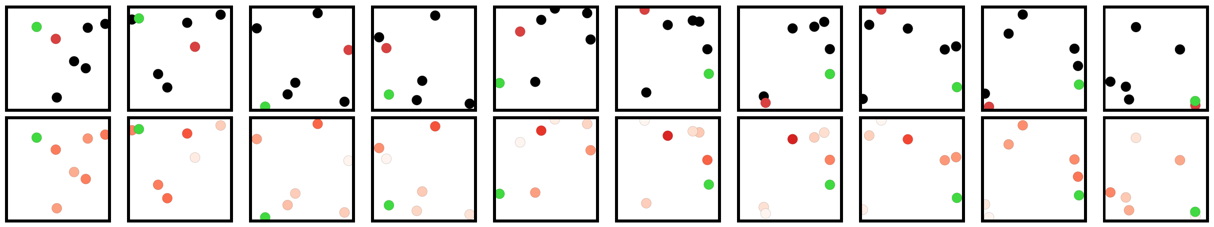
(c) Value function of distance by ToM

Figure 4.5: Learned zero-level value functions in the **fast**-thief setting. (a) and (b) shows the learned values on the mental state learned by ToM-gt and ToM, *i.e.*, the value of our agent that the thief thinks our agent is/is not the police. Our agent successfully learns that being recognized as a police has a lower value. (c) shows the final learned value function of the distance between our agent and the thief. Our agent learns that when the thief is very close, there is a high value to directly approach the thief. When the thief is far away, getting closer will not gain much value.

Figure 4.5 shows that our algorithm learns meaningful zero-level value functions for both the physical state and the mental state.



(a) An example episode played by MADDPG [15].



(b) An example episode played by ToM (our method).

Figure 4.6: Qualitative results in the **fast**-thief setting. Each group of two rows shows in the first row the ground truth state, and in the second row the beliefs of the thief along time (from left to right). The green one is the thief and the red one is the police. Darker colors indicate higher beliefs. Agents appear on the opposite side when they cross the boundary. (a) The thief recognizes the MADDPG agent as the police and escapes. (b) Our ToM agent successfully hide its identity and catches the thief.

CHAPTER 5

Conclusion

In this work, we propose a novel way to model the nested belief update, which alleviates the computation problem and provides an alternative way to learn other agents' models. We model the nested belief update by a Markov probability transition, leading to a linear transformation. The transition kernel is learnable and provides an efficient nested belief update. In the experiments we show that our agent is able to learn other agents' belief updates and intentionally change other agents' beliefs to achieve its goal. We believe this work interprets the belief update from a unique perspective and will benefit research in relevant domains.

Bibliography

[1] Chris Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B. Tenenbaum. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1:0064, 03 2017.

[2] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[3] Colin Camerer. *Behavioral game theory: Experiments in strategic interaction.* Princeton University Press, 2003.

[4] Prashant Doshi and Piotr J Gmytrasiewicz. A particle filtering based approach to approximating interactive pomdps. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2005.

[5] Prashant Doshi and Piotr J Gmytrasiewicz. Monte carlo sampling methods for approximating interactive pomdps. *Journal of Artificial Intelligence Research*, 2009.

[6] Prashant Doshi and Dennis Pérez. Generalized point based value iteration for interactive pomdps. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

[7] Prashant Doshi, Yifeng Zeng, and Qiongyu Chen. Graphical models for interactive pomdps: representations and solutions. *International Conference on Autonomous Agents and MultiAgent Systems*, 2009.

[8] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

[9] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[10] Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 2005.

[11] J Kiley Hamlin and Karen Wynn. Young infants prefer prosocial to antisocial others. *Cognitive development*, 26(1):30–39, 2011.

[12] Yanlin Han and Piotr Gmytrasiewicz. Learning others intentional models in multi-agent settings using interactive pomdps. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[13] Trong Nghia Hoang and Kian Hsiang Low. Interactive pomdp lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[14] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[15] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6379–6390, 2017.

[16] Yuyan Luo. Three-month-old infants attribute goals to a non-human agent. *Developmental science*, 14(2):453–460, 2011.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[18] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[19] Kristine H Onishi and Renée Baillargeon. Do 15-month-old infants understand false beliefs? *science*, 308(5719):255–258, 2005.

[20] Alessandro Panella and Piotr Gmytrasiewicz. Bayesian learning of other agents' finite controllers for interactive pomdps. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[21] Neil C Rabinowitz, Frank Perbet, H Francis Song, Chiyuan Zhang, SM Eslami, and Matthew Botvinick. Machine theory of mind. In *International Conference on Machine Learning (ICML)*, 2018.

[22] Simo Särkkä and Jouni Hartikainen. Infinite-dimensional kalman filtering approach to spatio-temporal gaussian process regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[23] Harm Seijen and Rich Sutton. True online td (lambda). In *International Conference on Machine Learning (ICML)*, 2014.

[24] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

[25] Ekhlas Sonu and Prashant Doshi. Scalable solutions of interactive pomdps using generalized and bounded policy iteration. *Autonomous Agents and Multi-Agent Systems*, 2015.

[26] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction.* MIT press, 1998.

[27] Richard S Sutton, Hamid R Maei, and Csaba Szepesvári. A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[28] Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 2009.

[29] Friedrich Burkhard Von Der Osten, Michael Kirley, and Tim Miller. The minds of many: opponent modelling in a stochastic game. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[30] Jörgen W Weibull. *Evolutionary game theory*. MIT press, 1997.

[31] Henry M Wellman and Susan A Gelman. Cognitive development: Foundational theories of core domains. *Annual review of psychology*, 43(1):337–375, 1992.

[32] Amanda L Woodward. Infants selectively encode the goal object of an actor's reach. *Cognition*, 69(1):1–34, 1998.

[33] Chongjie Zhang and Victor R Lesser. Multi-agent learning with policy prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.