Mobility Project Report: Modeling Others Using Oneself in Multi-agent Reinforcement Learning

Primož Prevc
University of Ljubljana
Ljubljana, Slovenia
primoz.prevc95@gmail.com

Igor Farkaš
Comenius University in Bratislava
Bratislava, Slovak Republic
farkas@fmph.uniba.sk

Matej Pecháč
Comenius University in Bratislava
Bratislava, Slovak Republic
matej.pechac@gmail.com

***Abstract*—In this project, we replicated a self other modeling mechanism, where reinforcement learning agents use their own neural network to simulate the behaviour of other agents and try to infer their goals. This mechanism is based on the mirror neuron system and simulation theory, which claims we understand the beliefs, goals and mental states of others by simulating those mental states in ourselves. We managed to replicate the goal inference mechanism on a simplified neural network architecture, but this process did not seem to increase the task performance of our agents. Further research is proposed to gain a better understanding of this process and its implication on theory of mind and cognitive science.**

I. Introduction

Reinforcement learning is an area of machine learning, where agents learn by interacting with their environment, trying to maximize the reward they receive over some period of time. Some approaches in reinforcement learning are similar to supervised learning, except that the data set is not provided by the programmer, but collected by the agent through interaction with its environment, and is usually represented as sets containing a certain state of the environment, the action taken by the agent, the next state, resulting from this action, and the reward the agent received for it. The agent then uses a batch of these collected experiences to optimize its policy, which defines which actions should be taken in a certain state of the environment.

While the mathematical foundations of reinforcement learning come from areas such as dynamical programming, cybernetics, and control theory, there have also been many reciprocal connections with cognitive science and psychology, with many core algorithms of reinforcement learning being inspired by biological learning systems (Sutton & Barto, 2018). The main idea behind reinforcement learning is almost analogous to operant or instrumental conditioning in behavioral psychology, with the reward signal performing a similar role as pleasure and pain in humans and animals. While behaviorism has largely been abandoned due to its difficulties with explaining the more complex aspects of human cognition, there has recently been an increasing interest in embodied approaches to cognitive science, which emphasize the role of sensorimotor interaction with the environment as a constitutive factor of cognition. Furthermore, these embodied approaches also draw on the theoretical foundations of cybernetics and dynamic systems theory, so it seems that an interdisciplinary integration of reinforcement learning and embodied cognitive science would be feasible and could provide useful insights for both of the fields.

A. Theory of mind

In multi-agent settings, an agent's environment also contains other agents, which might have different goals and policies that guide their behaviour. If the reward that an agent receives also depends on other agents, then the agents would benefit from trying to infer the goals and beliefs of other agents, in order to predict their behaviour. In psychology, the ability to infer the mental states of others is called the theory of mind. To some degree, this ability is already present in infants and it develops through childhood to include not only inferring others' desires and beliefs, but also realizing we know something another person does not, or that someone has a false belief about something (Wellman, 2018).

Rabinowitz et al. (2018) used reinforcement learning to design a Theory of Mind neural network, which was able to infer different agents' characteristics and beliefs, and use them to predict their behaviour. The model was even able to recognize that agents with imperfect information sometimes had false beliefs, and predicted they would act according to them.

B. Simulation theory

The previously described model could be said to represent an approach called the *theory-theory of mind*, which claims that we possess some sort of model of human psychology, which we use to infer mental states of other people. This model is supposed to be improved over time in a similar way that scientists improve scientific theories, by collecting more data to be able to make more accurate predictions. An alternative approach is called the *simulation theory of mind*, which claims we understand the mental states of others by taking their perspective and simulating those mental states in ourselves. This theory is based on the discovery of mirror neurons, which are active both when a certain action is executed and when the same action is observed. This action execution and observation matching mechanism is hypothesized to be not only a basis for imitation learning and understanding others' motor actions, but also an integral part of our capability to infer and understand others' mental states.

Reileneou et al. (2018) used this idea to develop a reinforcement learning model, which uses itself to model the behaviour of other agent's and tries to infer their goals. Agents performing different kinds of competitive and cooperative tasks tried to infer the goals of another agent by predicting what they would do themselves, if they were in the other agent's situation, using the same neural network that guides their own behaviour. The agents can then observe how the real actions of other agents differ from their predictions of other agents' actions, and use this difference to adjust their prediction of other agents' goals.

C. Self-other modelling

The agents' policy is parameterized by a neural network. The input to the neural network consists of the agent's own goal, the goal of the opponent, and the representation of the current state of the environment. Agents can have different task goals, which are provided by the environment at the start of each episode. The output of the neural network is a vector of probabilities, corresponding to each of the available actions. The network is used in two different ways, the acting mode and the inference mode. The agents take turns in taking actions during one episode of the task, so at every episode step the acting agent uses its network in the acting mode, while the other, observing agent uses its network in the inference mode, to try to infer the goal of the acting agent (see Fig. 1 for a schematic representation of the way the network is used in the two modes.)

In the acting mode, the agent uses the *true self goal*, that is the goal that was provided by the environment in that episode, and *predicted other's goal*, which is the current prediction of the opponent's goal, which has been determined by the goal inference process so far. The action to be taken is then sampled from the probability distribution, resulting from the feedforward pass of this neural network.

In the inference mode, the agent puts itself in the position of its opponent, and simulates what it would do in that situation. To do this, it uses the same neural network, but the position of the inputs is switched around. In the place of its own goal, it puts the current prediction of the opponent's goal, or the predicted self goal, while in the place of the opponent's goal it puts the true other's goal, which is it's own goal, provided by the task environment. The representation of the environment used in the input is also taken from the perspective of the opponent. The agent then uses this input in it's feedforward network pass, so it simulates what it would do, if it were in the opponent's situation, given the current prediction of the opponent's goal. It then observes the true action, performed by the opponent. Then it computes the cross entropy loss between the probability distribution, resulting from its inference mode feedforward pass and the true other's action, performed by the acting agent. If the true action is incongruent with the predicted probability distribution, the loss will be large, otherwise it will be smaller. The loss is then backpropagated through the network. However, unlike in the usual neural network backpropagation, the loss gradient is not used to adjust the parameters of the network itself, but is used to adjust the input of the network, specifically the predicted goal. By repeating this step several times throughout the episode, the process of gradient descent adjusts the prediction of the opponent's goal in a way that makes the resulting action probability distribution congruent with the actions performed by the acting agent.

At each episode steps several inference steps are performed. Since the predicted other's goal is a probabilistic representation, while the true other goal is a discrete one-hot encoded representation, we use the Gumbel-Softmax reparameterization trick (Jang et al. 2016) to create a differentiable sample from the probabilistic representation, similarly to Railaenu et al. (2018). Predicted other's goal is represented as a vector

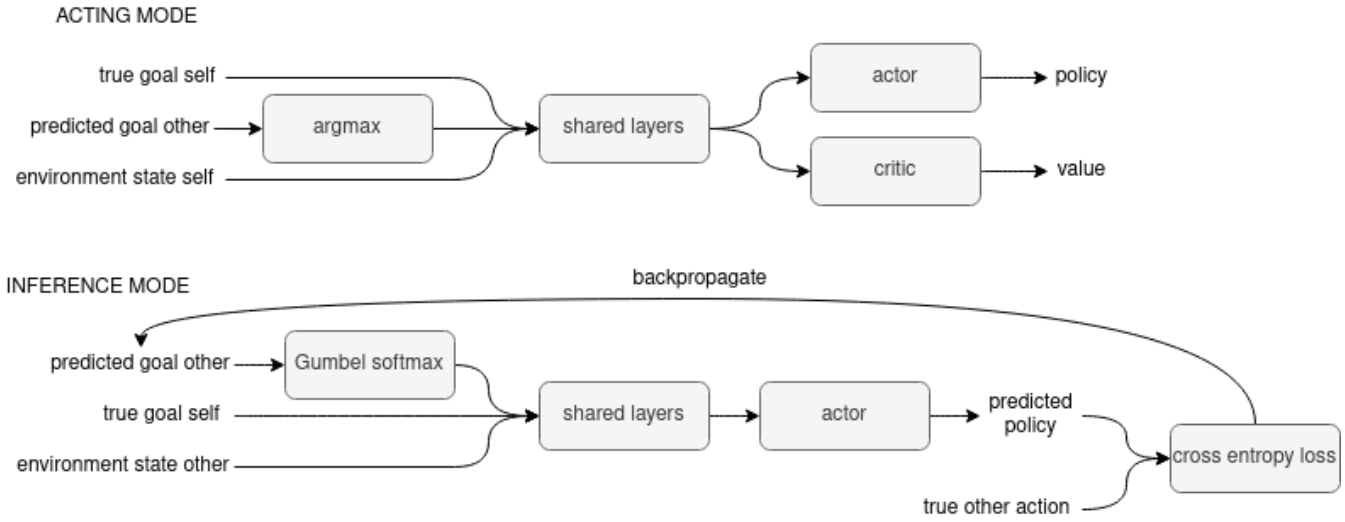

Fig. 1. Schematic diagram of the self-other modeling mechanism

of unnormalized log probabilities. This is because the input to the Gumbel-Softmax function we used is in the form of unnormalized log probabilities. In the acting mode, an *argmax* function is used to transfrom the goal into a discrete one-hot representation, and the *argmax* of a vector of probabilities and log probabilities is the same.

II. Method

A. Coin game task description

The task that was used to test this model was a simple coin game, which was also used in the original in Raileanu et al. (2018). In this task, the two agents and twelve coins of three different colors are placed in randomized positions in a 8x8 gridworld environment at the beginning of each episode (see Fig. 2 for an example). For every agent a different random goal is chosen, which determines which color of the coins they should pick up. The agents then have ten steps each to pick up as many coins as they can, with the possible actions at each step being moving up, down, left or right for one field, or to stay still. The coin is picked up automatically, when they move onto a field where the coin is located. They receive rewards based on the amount of collected coins corresponding to both their own color and their opponent's color, and they receive a penalty if they pick up a coin that is neither their own or their opponent's. To maximize the received reward, the agents must therefore infer the goal of their opponent as fast as possible, so they can start to pick up their coins as well.

B. Network architecture

In our implementation of the model, we made certain changes from the model described in the original paper. In their model they used LSTM, while we used a basic fully connected neural network, as we believed that the goal inference mechanism should also work with this kind of network, so we wanted to make the most basic implementation of the mechanism. Since this kind of a model architecture is memoryless, which means it only considers the current state of the environment when choosing its action, we distribute the reward at every episode step, so as soon as a certain coin is picked up, while in the original paper they distribute it at the end of each episode. Also, in their task, the reward is shared between the agents, so both agents receive a reward which results from the coins they collectively picked up, while in our task, each agent only receives the reward based on its own collected coins. This makes the task competitive instead of cooperative as it is in the original paper. This does not significantly affect the behaviour of the agents, but it is something to keep in mind when comparing the received rewards between the two papers. The last difference is the training algorithm used to update the model parameters between the episodes. They used the A3C algorithm while we used PPO. This was mostly done to avoid the additional complexity brought by the parallel and asynchronous nature of A3C, as well as the relative ease of tuning of the PPO algorithm. Since they are both policy gradient methods, we did not expect that the change of algorithm would make a significant difference to the model performance.

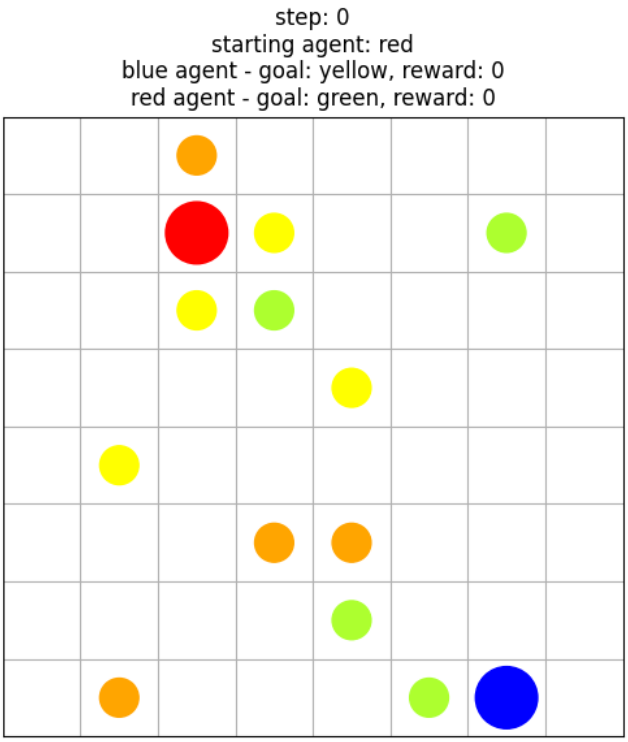

Fig. 2. Example of a randomized starting position of the coin game task.

Similarly to Raileanu et al. (2018), we first developed two baseline models that were used to assess the effect of the goal inference process on the agent's performance. In the first baseline, true other goal (TOG), the agents received the true goal of their opponent as the input to the model. This baseline represents the upper boundary of the agent's performance on a particular multi-agent task. In the second baseline model, no other model (NOM), the agents do not use any information of the other agent's goal in their model. Finally, we added the goal inference process to the TOG model, so instead of the true goal of the opponents, the agent used their prediction of the opponent's goal in their model, thus making a self-other model (SOM). We also used the two baseline models to find an appropriate network architecture and training hyperparameters, since the addition of the goal inference process adds significantly to the computational complexity of the task.

C. Training parameters

The neural network architecture used was an actor-critic with two shared hidden layers with 256 neurons, as well as a separate layer for the actor and critic with 128 neurons. We used a PPO algorithm implemented by Matej Pecáč for his research on intrinsic motivation in reinforcement learning. The hyperparameters used were: gamma = 0.99, beta = 0.01, actor loss coefficient = 0.5 and a learning rate of 0.001.

The input layer had a size of 390, and consisted of a one hot encoding of the goals of both agents, as well as an

environment representation, where each field was encoded with a one hot encoding with size of six, representing either an empty field, position of self (the agent whose perspective the environment is observed from) and the other, as well as each of the different class of coins.

III. Results

A. Baseline models

Fig. 3 shows the results number of coins picked up and the total collected reward of the TOG and NOM baselines. Since the results vary a lot from episode to episode, they were smoothed with a rolling average with the window size 10^4. The reward is calculated in the same way as in Raileanu et al. (2018), according to the following formula:

$$R = (n^{self}_{Cself} + n^{other}_{Cself})^2 + (n^{self}_{Cother} + n^{other}_{Cother})^2 - (n^{self}_{Cneither} + n^{other}_{Cneither})^2$$

In the TOG baseline, self and other's coins are picked up with the same frequency, while in the NOM baseline, the agents only pick up the coins corresponding to their own goal. In both cases, the agents sometimes also pick up coins that correspond neither to self or other's goal. This could mean that their learned behaviour is not completely optimal. However, since the number of steps in each episode is limited to 10, it is sometimes better to pick up an incorrect coin, if it means that an additional correct coin can be collected, since the reward increases quadratically with the amount of coins of the same color.

B. Self other modeling

Fig. 4 shows the number of collected coins of the SOM model, and the success rate of the goal inference process during training. At the beginning of training, the goal prediction rate is the equal to random guessing, with all three goals being predicted with the same probability. After around 3-4 million episodes of training, the agents correctly predict the other agent's goal with almost a 70% accuracy, which is similar to the original paper. However, the agents do not seem to be using this information in their behaviour, as they still continue to collect only the coins corresponding to their own goal.

IV. Discussion

While the agents learned to infer the opponent's goal with an average accuracy of 60% after approximately 2 million episodes, they still predominantly picked up the coins corresponding to their own goal and not the opponents. In the beginning of the training process, the predicted opponent's goal was essentially random, so the agent's have learned to ignore this input. The neural network weights in the first fully connected layer, that are connected to the part of the input that represents the opponents goal, are reduced to almost zero, so these inputs do not contribute almost anything to the output of the network. As the goal prediction process gets more and

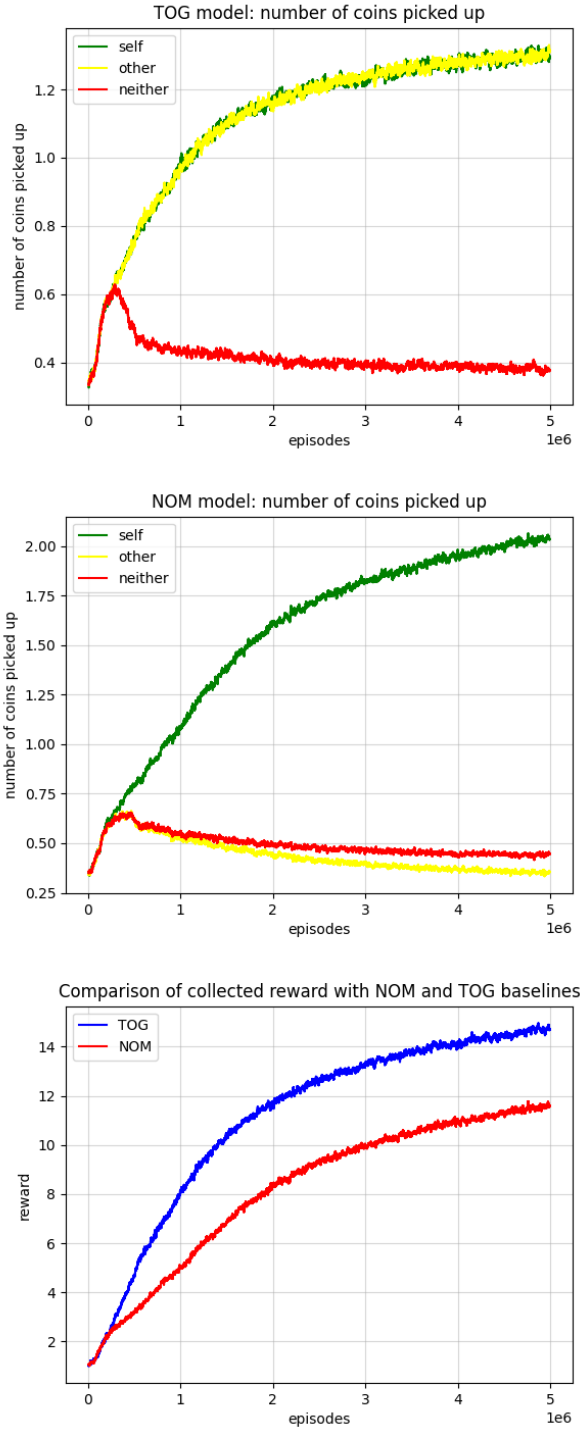

Fig. 3. Number of coins picked up with the TOG and NOM baselines.

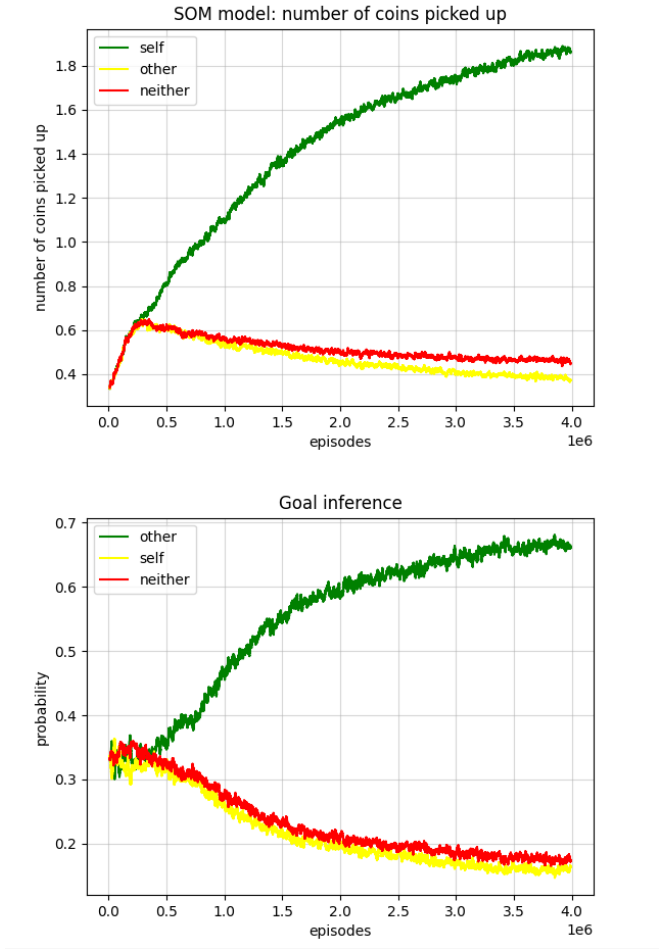

Fig. 4. Number of coins picked up with the SOM model and goal inference performance during training.

more accurate however, these weights remain unchanged, even though they should be adjusted by the training algorithm. We are not quite sure why these weights were not being updated during the later stages of the training, but we have some potential solutions that we intend to test out with further research on this topic.

One potential solution could be to increase the magnitude of the one hot encoded value of the predicted goal, as it gets more and more accurate. This would increase its relative contribution to the output of the network, which could perhaps make the weights, corresponding to this input, more likely to be adjusted during training. Another option would be to separate the goal representations from the environment representation. The first few layers of the network would encode the environment into a smaller latent representation, with fever neurons, and the goals would then be concatenated into this more compact representation, which would also increase their relative importance.

There are also other avenues of further research that could be pursued. One of them would be comparing different network architectures. For example a convolutional neural network could provide a more efficient environment representation, since it implicitly represents some of the two dimensional spatial relations of the environment and the task. The mechanism could also be extended to a situation with more than two agents, which would also allow us to investigate more complex interactions between the agents. There could be different populations of agents which interact with each other during training. We could see if the goal prediction performance differs between the agents within the same population and between different populations. These investigations could give us a better understanding of this process and how it relates to the mirror neuron system and simulation theory.

References

[1] Gallese, V., & Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. Trends in cognitive sciences, 2(12), 493-501.
[2] Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144.
[3] Raileanu, R., Denton, E., Szlam, A., & Fergus, R. (2018, July). Modeling others using oneself in multi-agent reinforcement learning. In International conference on machine learning (pp. 4257-4266). PMLR.
[4] Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. A., & Botvinick, M. (2018, July). Machine theory of mind. In International conference on machine learning (pp. 4218-4227). PMLR.
[5] Rizzolatti, G., & Craighero, L. (2004). The mirror-neuron system. Annu. Rev. Neurosci., 27, 169-192.
[6] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
[7] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
[8] Wellman, H. M. (2018). Theory of mind: The state of the art. European Journal of Developmental Psychology, 15(6), 728-755.