Re-Inforcement Learning In Multi Agent System With Transfer Learning

Amita Dhariwal1, Lata Bharti2 and Dr. Sandhya Tarar3
1,2,3School of Information and Communication Technology, Gautam Buddha University, Gr. Noida, INDIA

ABSTRACT
Transfer learning is the process that uses the previous task knowledge in the current task to improve the performance of the new task. In reinforcement learning, the agent requires training from the task. Transfer learning use with the reinforcement learning to improve the performance of agent. Transfer learning method is mainly applied on single agent reinforcement learning algorithms. We use the better algorithm for transfer learning with reinforcement learning on multi agent domain. Domain is a real time strategy game (RTS). In multi agent one agent is a cooperative agent and another is competitive agent. In reinforcement learning the agent give limited feedback or information to agent. The main role of transfer learning is that the learning in one task can help to improve performance in another task. In multi agent framework the advice given by agent in one stage can be used in another stage of the system. Our experiment conduct on real time strategy game setup. The result shows that Bias Transfer reduces the training time in the target task and improves asymptotic performance.

Keywords---- Multi-agent systems, Reinforcement learning, Transfer learning, Soccer game.

I. INTRODUCTION
In the reinforcement learning many problems require a huge amount of training time to solve the problem. When transfer learning use with the reinforcement learning in the same problem training time is reduce. Till now several methods have been proposed to solve these types of problems to reduce complexity. Most of the proposed methods are applied on single agent problems. In multi agent system the agents communicate to each other to achieve the same goal. In multi agent system the agents choose the best action among several actions to accomplish the same goal. In this paper we present the method that uses the transfer learning with reinforcement learning in multi agent system. We represent this method in multi agent reinforcement learning (MARL). The proposed method is based on Bias Transfer, which is applied on MARL with transfer learning.

The basic idea about bias transfer method is to use joint policies; means the agents learn in the source task can be apply in the target tasks that accomplish the same goal. In the bias transfer method we use Joint Action Policy as the basic learning algorithm. The proposed method can be used in multi agent domain applications. The proposed method checked on multi agent soccer domain. The result shows that the proposed method can reduce the learning task in the domain to achieve the goal and also improve the asymptomatic performance.

II. TRANSFER LEARNING
Transfer learning is that learning in which the agents learn in the one task as the learning purpose and use in another tasks which is related to same domain. Many issues occur in the transfer learning method like how the tasks are related to each other and how these tasks are different and which part of the knowledge should be transfer or not. The tasks of the same domain may have different state space with fixed variables [7] or even different state variables [5]. Many methods have been discussed in which tasks in the different state and action space and also in the transition function and reward [12]. These proposed methods use inter tasks mapping means they use relation in the source task to other tasks. Mapping between source and target tasks can be shown as the (xa, xb) where

\[ x_a(s) : S_{target} \rightarrow S_{source} \]
\[ x_a(a) : A_{target} \rightarrow A_{source} \]

A survey found on transfer learning in the single agent reinforcement learning [9]. The transfer knowledge between the agents in the tasks may be low in which
tuples as the form a, r, s′ [6] and value function [12]. The knowledge in which higher may include these tuples like a method for single agent problem solving. In multi agent the agents learns from other agents in the learning point of view. A method used in the transfer learning in single agent learning is not equal to the multi agent learning. Multi agent in any domain has been applied is a major task. In this section we will provide several issues that are arise in the multi agent reinforcement learning domain. In the context of learning in the multi agent domain is the specifically a effect transfer learning, which has some restriction. First of all we consider the homogenous agents that mean agents have higher degree of similarities among the actions to achieve the target. We also assume that the agents may be competitive, means agents does not learn the behavior of the opponent agent in the domain. Agent homogeneity may be high restrictive; tasks with heterogeneous agents can be viewed as having many different classes of mutually same agents; then transfer would generally still take place between these same agent classes across tasks, the transfer task in this case could be seen as a series of parallel homogeneous transfers.

Inter task Mappings across Multi agent Tasks

Inter task mappings in single agent tasks use the very similar states and actions between the source and goal tasks. A difference in the multi-agent domain is that the learned knowledge for each task is distributed among other agents, which means that the mapping factor for the goal task have to be defined each agent. We propose a function defined for agent i that map the joint actions of n-agent task to those of an m-agent task below:

\[ \chi_{i,j} : A_1 \times \ldots \times A_n \rightarrow A'_1 \times \ldots \times A'_m \]

where \( J_k = A_1 \times \ldots \times A_k \)

Correspondingly a map function that maps links between tasks can be defined per agent. Although states have the same meaning in multi agent tasks as in a single agent one, they can include parameters that are associated with a specific agent. Since it is useful in a multi-agent setting to make this distinction, we denote these parameters as \( s \) and as \( s' \) the rest of the state variables in the multi agent j tasks. The proposed form of such a map function for agent i is:

\[ \chi_{i,j} : S_n \rightarrow S_m(s) : S_n \rightarrow S_m \]

where each state \( s \in S_n \) and \( s' \in S_m \) of the target and source tasks correspondingly has the form action subsets or shaping rewards [5]. Madden and Howley has been proposed

\[ s: <s,agent_1,\ldots,agent_n> \]

\[ s': <s',agent_1,\ldots,agent_m> \]

The source and target tasks may have various action and state variables combination and these can be mapped using with the same techniques which one would use in a single agent task. There are a various techniques to define these mappings, especially when goal specific tasks are taken into account. A factor to representation of an agent in a one task is considered equivalent to an agent representation in the goal task. In many situations this mapping corresponds to that in which where each agent is thought to retain its identity over the two different domains. But it may be possible for a single agent to be mapped to the states and actions of various agents. Accordingly, we propose an approach.

Static agent mapping implements a one-to-one mapping between agents that is constant. This approach effectively ignores the presence or absence of actions the extra agents. This indicates that the chosen set of extra agents remains the same for all states and joint actions1. For example, shown below are functions defined for one Agent that map a three agent task to a two agent one, ignoring other Agent:

\[ \chi_{1,j} : S_n \rightarrow S_m (<s'_\text{target}, agent_1, agent_2, agent_3>) = <s'_\text{source}, agent_1, agent_2> \]

where \( a_{ij} \) is the jth action of the ith agent in the domain. It is important to show that these functions are simplified for demonstrative purposes; they make assumption that \( s'_\text{target} \) can be mapped directly to \( s'_\text{source} \) and that each agent has the same associated state variables and actions across tasks in the same domain. It is also important to keep in mind that these functions are defined for each agent.

\[ \chi_{1,j} : J_n \rightarrow J_m (<a_1,1, \ldots, a_1,i, a_2,1, \ldots, a_2,j, a_3,1, \ldots, a_3,k>) = <a_1,1, \ldots, a_1,i, a_2,1, \ldots, a_2,j> \]

where \( a_{ij} \) is the jth action of the ith agent in the domain. It is useful in a multi-agent setting to make this distinction, we denote these parameters as \( s \) and as \( s' \) the rest of the state variables in the multi agent j tasks. The proposed form of such a map function for agent i is:

\[ \chi_{i,j} : S_n \rightarrow S_m(s) : S_n \rightarrow S_m \]

where each state \( s \in S_n \) and \( s' \in S_m \) of the target and source tasks correspondingly has the form:

\[ s: <s,agent_1,\ldots,agent_n> \]

\[ s': <s',agent_1,\ldots,agent_m> \]

The source and target tasks may have various action and state variables combination and these can be mapped using with the same techniques which one would use in a single agent task. There are a various techniques to define these mappings, especially when goal specific tasks are taken into account. A factor to representation of an agent in a one task is considered equivalent to an agent representation in the goal task. In many situations this mapping corresponds to that in which where each agent is thought to retain its identity over the two different domains. But it may be possible for a single agent to be mapped to the states and actions of various agents. Accordingly, we propose an approach.

Static agent mapping implements a one-to-one mapping between agents that is constant. This approach effectively ignores the presence or absence of actions the extra agents. This indicates that the chosen set of extra agents remains the same for all states and joint actions1. For example, shown below are functions defined for one Agent that map a three agent task to a two agent one, ignoring other Agent:

\[ \chi_{1,j} : S_n \rightarrow S_m (<s'_\text{target}, agent_1, agent_2, agent_3>) = <s'_\text{source}, agent_1, agent_2> \]

where \( a_{ij} \) is the jth action of the ith agent in the domain. It is important to show that these functions are simplified for demonstrative purposes; they make assumption that \( s'_\text{target} \) can be mapped directly to \( s'_\text{source} \) and that each agent has the same associated state variables and actions across tasks in the same domain. It is also important to keep in mind that these functions are defined for each agent.
where transfer learning from a task with two other cooperative agents leads to a three agent one can have two another outcomes.

**Dynamic agent mapping** is the mapping in which agent’s action combination should remain the same as the requirement for all states and joint actions. This means that the agents do not retain an identity across the two other tasks. There are two ways to implement such a mapping function. For example, From the viewpoint of agent 1, such mapping functions for a three agent representation mapped to a two agent one using distance as a criterion would be:

\[ d(x_1, x_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \]

Where \( d(x_1, x_2) \) is the distance between agents \( x_1 \) and \( x_2 \) in the current state. The main difference in this case is that the action map function is a function of the current state \( s \) being mapped, as in this case it depends on its properties (i.e. the agents’ current coordinates).

**Level of Transferred Knowledge**

A Multi agent system is that in which the acquired knowledge is distributed among agents instead of residing in a single agent. This can be a major task for transfer methods, since there is no simple way to deal with multiple sources in the normal case. We select to transfer the learned joint policy in order to avoid this issue, since we can use this unified source of knowledge to transfer to each other agent. The trade-off to be made here is that some knowledge that could benefit the goal task is ignored, such as the values of suboptimal actions.

**Method of Transfer**

In the level of knowledge transferred, we must also decide how to incorporate this useful knowledge in the target task’s learning algorithm. The usual criterion for convergence in single agent system algorithms is to provide a correct estimate of the state and action value function that can be used to estimate the optimal policy.

We propose a method for transfer that incorporates the transfer knowledge as bias transfer function values in the initial action state value function. Since proof of convergence does not rely on the specific initial values of this function, we are essentially using MARL algorithm as the base. We proposed algorithm as a Bias transfer method that does not affect the convergence of the underlying reinforcement learning algorithm. Previous research in biasing the initial Q values [7] generally avoids defining the specific intervals that the bias parameter should lie within that interval. This is justified, since an optimal bias parameter value relies on the specific properties of the Q function that is being estimated in the first place of the agents. Intuitively, we seek a value enough such that it will not be overcome by smaller rewards before the goal state is reached within a few times, and low enough to not interfere with learning in the later stages. Our experiments have shown that for most problem a relatively small bias (e.g. \( b = 1 \) when \( R_{max} = 1000 \)) usually has better results and performance will start to drop as this value is increased. Using a bias value \( b \), Algorithm shows the pseudo code for the generic multi-agent transfer.

**Algorithm BIAS TRANSFER** for agent \( i \)

1. for all states \( s \) in \( S_{target} \) do
2. for all joint action vectors \( \alpha \) in \( A_1 \times ... \times A_n \) do
3. \( Q_{target}(s, \alpha) \leftarrow 0 \)
4. if \( \chi_i, A_n \rightarrow m(\alpha - n) = \pi_{source}(\chi_i, S, n \rightarrow m(s)) \) then
5. \( Q_{target}(s, \alpha) \leftarrow b \)
6. end if
7. end for
8. end for

Q-value reuse adds the Q-values of the source task directly to the Q-values of the goal task. In this algorithm, the new Q-values are defined as:

\[
\begin{align*}
Q_{i, target}(\cdot, \cdot) & \leftarrow Q_{i, target}(\cdot, \cdot) + Q_{source}(i, S, n \rightarrow m(i), i, A_n, n \rightarrow m(\alpha - n))
\end{align*}
\]

However, unlike the previous method that is only invoked before learning, transfer here takes place during the execution of the target task and becomes a part of the learning algorithm. A significant difference in this case is that one would have to choose which \( Q_{source} \) to use. This could be the Q function of an individual agent in the source task such as an average from all agents.

**IV. EXPERIMENT DOMAIN**

In order to evaluate the proposed methodologies we used the soccer domain. The domain is a discrete grid-world where there are two types of agents one is predators and another prey. The goal of the predators is to keep the goal against preys as fast as possible. The grid is toroid and fully observable, which means that the predators receive accurate information about the state of the environment.

The learning environment in all cases was a 5 × 5 grid, where the current state is defined by the locations of the prey and the other predators. The agents can choose their next move from the action set \( A = \{ \text{NORTH, SOUTH, EAST, WEST, NONE} \} \) in which NONE means that they remain in their current position. States in this condition include the \( x \) and \( y \) coordinates of the prey and the other predators, relative to the current predator, so a state from the viewpoint of predator \( A \) in a two agent world with another predator \( B \) would be of the form:

\[
s = (\text{prey}_x, \text{prey}_y, B_x, B_y)
\]

In all cases for both source and goal tasks the MARL algorithm used is joint action learning (JAL). The
exploration method used is Boltzmann exploration, where in each state the next action is chosen with a probability of
\[ \Pr(a_i) = \frac{e^{Q(s,a_i)/T}}{\sum_{j=1}^{n} e^{Q(s,a_j)/T}} \]

Where the function is calculate of the maximum value of all possible joint Q actions given an agent’s individual action. T is the temperature parameter, where \( N_s \) is the number of times the state was visited before and \( C_t \) is the difference between the two highest Q-Values for the current state of the agent. Boltzmann exploration was fully used in the single and multi-agent version of the task, but in the three agent version it was more practical to use in 10% of the steps, making it the exploration part of an e-greedy method where \( Q = 0.1 \). For all experiments we used a constant learning rate \( a = 0.1 \) and a discount factor \( \gamma = 0.9 \). When BITER is used, the bias parameter is \( b = 1 \). The rewards given to each individual agent were \( r = 1,000 \) for goal the prey, \( r = -100 \) when collision with another agent occurs, and \( r = -10 \) in all other states. For experiment trials were conducted. The results that we present are averaged over these repetitions.

V. RESULTS

For each experiment, we record the performance times in terms of capture better and non-transfer better the results do not include the learning time of the source task as it is typically an order of magnitude less than the target task’s. The first of all the soccer experiments involve two tasks of the team capture game, with one, two and three predators respectively. Additionally, we use the dynamic mapping method for all transfer procedures.

The first transfer case focuses on the out of circle team capture task, where we applied our proposed transfer method using a single-predator capture task as source. In this simple case, the learned policy of the source task is used to bias the initial Q function of the target task. The learning time for the source task is approximately 200 episodes, or about 1200 cycles in total. Since the size of the state and action space is relatively small, it can be assumed that the source task’s learned policy is optimal. In this case each agent in the target task begins with a policy that is biased towards the learned policy from the single-agent task.

Figure 2 represents the results of BITER compared to the non-biter algorithm capture time

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>Performance of agents with BIAS transfer in Robo Cup soccer game.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agents</td>
<td>Players</td>
</tr>
<tr>
<td>Algoritms</td>
<td>Non Transfer</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, we present method to reduce the amount of training time for RL with the help of transfer learning. The main idea was to build extensive knowledge from few experiences. This is crucial for the application of RL methods to real-world scenarios. We use imitation to replace the random exploration of the large state and action space with a guided exploration. In our approach, the agent has full access to experiences of a teacher, which has the same state and action space and gets identical rewards. Perceptions, actions, and rewards of the experienced agent are stored and can be accessed and reused later for the same type tasks. Similarly, own experiences are stored and reevaluated later. This
basically reduces the training expenses. We let the agent repeatedly reprocess past experiences to avoid this problem. In addition, the quick generalization of similar situations while preserving the possibility to distinguish between various situations, essentially contributes to the acceleration of the learning process. As the experimental results show, fundamental soccer skills can be learned using RL in simulation. The approach also works with a real humanoid robot on the soccer field. The given task is accomplished quickly and reliably. Although the training with the real robot requires more time than the training in simulation, it stays within limited period.

REFERENCES


