Recent Developments In Multi-Agent Reinforcement Learning

Anonymous CVPR submission

Paper ID ****

Abstract

The field of reinforcement learning has seen massive success in recent years, surpassing human performance in strategic games like Starcraft 2 and Go. A number important applications require multiple agents interacting with each other, deriving complex behavior through cooperation, competition, or a mix of both. We look at 2 different works which propose solutions for learning in a multi-agent environment: Lowe et al.[1] introduce an actor-critic model which is augmented to perform in a multi-agent setting, while Liu et al.[2] explore how cooperative behaviour emerges through competition in football-playing agents. Finally, we present Google Football, a novel football-based environment for reinforcement learning, developed by the Brain Team at Google Research (Kurach et. al[3]), which aims to bring a strong aid to future researchers of the field.

1. Introduction

In reinforcement learning, agents interact with the environment through actions, causing it to reach a new state, and the agent to receive a reward based on this new state. The goal is to maximise the long term expected reward. Modern research in the field of multi-agent systems has centered on reinforcement learning, where games have proven suitable for developing techniques for agents to co-evolve. In competitive games, simple rules give rise to complex behaviour as a result of competition. Such a game is the world's most popular sport: football (a.k.a. soccer). Agents have to learn to control their player, cooperate with their teammates (for example by passing) and score against the opponent.

1.1. Background

We consider a multi-agent extension of a Markov Decision Process called a Partially Observable Markov game, which consists of a set N of agents, a set S of states which encapsulates the environment and the position of each agent, a set A of actions that agents can take, and a set O of observations that each agent has. At every step, each agent uses a stochastic policy pi to choose and action and produce the next state, based on which it receives a reward. It's goal is to maximise the expected long term reward, often discounted so that the reward doesn't diverge.
Q-learning uses an action-value function for a policy pi, Q(s,a) which models how the long-term reward is going to look like if the agent following the policy pi takes action a at state s. Deep Q-Networks learn the action-value function by the means of a deep neural network, which tries to minimize a loss function. This method often uses a replay buffer D, consisting of tuples (s,a,r,s'), where s' is the state reached if action a is taken in state s. This cannot be used since policies of agents change during training in a multi-agent setting, thus making the environment non-stationary and violating the Markov convergence property.

1.2. The problem

Traditional reinforcement learning methods like Q-learning and policy gradient have yielded poor results in practice when used to train a single agent separately in a multi-agent setting. A reason for this is the environment becomes inherently non-stationary when the policies of the agents change during training, which limits the use of a replay buffer. In the case of policy gradient the variance increases drastically with the number of agents. Most previous studies are cooperative and assume that agents make actions to improve a collective reward (like optimistic q-function), or, in the case of parameter sharing, require agents to have the same structure. These assumptions fall in competitive or mixed settings.

1.3. The solutions

We look at 2 different deep reinforcement learning approaches that overcome the problems presented above: Lowe et al[1] propose an actor-critic variant where the critic is augmented to hold approximations of the policies of other agents; Lie et al[2] use single-agent reinforcement learning with population training and co-play.

2. The methods

2.1. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

Since we can't use an experience replay buffer which is crucial for stabilizing deep Q-learning, and policy gradient's variance explodes as the number of agents increases, [1] introduce a general-purpose method which doesn't assume a model of the environment or how the agents communicate, and works in both cooperative and competitive scenarios. This is done using an actor-critic framework with centralized learning and decentralised execution. By augmenting the critic with an experience replay buffer for every agent, it allows policies to train with additional information, and only use local information at execution time. A centralised action function is used that takes as input the actions of all agents and returns the Q-value for agent i. Thus, the environment becomes stationary even though policies change. An agent holds approximations of the policies of other agents, which are learnt. In practice, this approximation was found to be very close to the actual value, but without the computational overhead. To prevent overfitting another agent's policy, [1] split a policy into sub-policies, and at each episode select one at random for each agent to execute, maintaining a different replay buffer for each sub-policy. Results are measured against decentralized learning methods using 3 defferent scenarios:
Physical deception - N agents cooperate to reach a single target landmark from a total of N landmarks. They are rewarded if for each landmark if any agent is close to it. An adversary also desires to reach the target landmark; but it does not know which of the landmarks is the correct one. Thus the cooperating agents, who are penalized based on the adversary distance to the target, learn to spread out and cover all landmarks so as to deceive the adversary.
Predator-prey - in this variant of the classic predator-prey game, N slower cooperating agents must chase the faster agent in an environment filled with random obstacles. The agents are rewarded every time they touch the adversary, who, in turn, is penalized.
Covert communication - This is an adversarial communication environment, where a speaker agent has to communicate to a listener agent who has to reconstruct the message. Another adversary agent is listening in the channel, so the speaker agent has to come up with a randomly generated key to encript the messages, only known to the listener agent.

2.2. Emergent Coordination Through Competition

Lie et al.[2] propose a decentralised, population-based training algorithm, where subsets of agents from the population are selected to play against each-other. Each agent learns individually off-policy with reccurent memory and decomposed shaping reward channels. Agents start with random behaviour, and over time learn simple ball-chasing, and eventually show signs of cooperation. Agents are evaluated according to a fitness function, where weak agents' hyperparameters are inherited from the network, or from stronger agents, in addition to a mutation.
Stochastic Value Gradients are used as the reinforcement learning method: an actor-critic model policy gradient which estimates gradients of the objective policies and uses a differentiable Q-critic learnt using experience replay. Since other agents are modeled as part of the environment, and thus not directly recognized by an agent, [2] use a recurrent Q-critic ti enable the Q-function to learn other players' behaviours and better estimate the correct value of the current game. Reward shaping is an effective technique for incorporating domain knowledge into reinforcement learning. Since this is a difficult task, [2] use automatically evolving shaping rewards based on match results. Evaluation is done using a game theoretic aproach - Nash-Averaging Evaluators are used since we want invarience to redundant agents (multiple agents with similar strategies should not bias the ranking).

2.3. Google Research Football: A Novel Reinforcement Learning Environment

Rapid advancement in reinforcement learning has lead to the increasing need of environments like video-games. Google Football is a 3D physics-based learning environment with support for multi-agent settings and base-lines for 3 popular algorithms in the field. It's based on the game of football, and offers many features the game has, like 3 types of passing, side kicks, corner kicks, yellow and red cards, offside, hands, and penalties. Each player has statistics, like speed and accuracy, and get tired over time. 3 different aproaches are offered for the representation of the state:
1. Pixels - image corresponding to game state, that includes a scoreboard and a minimapm with relative player positions.
2. Super minimap - 4 72x96 matrices with information about team 1, team 2, the ball, and the active player.
3. Floats - 115 dimensional vectors summarising various aspects of the game: player coordinates, ball position and direction, active player.
Rewards are done using scoring (+1 point for the team who scores and -1 for the other), and using checkpoints that leverage domain knowledge (a good position to be is in the adversary's side of the field). Agents are rewarded when they advance deeper towards the adversary's field for the same time.
Thus, Google Football is looking to be a promising aid to future researchers in reinforcement learning, as it is open-source and has different trained checkpoints, while enabling support for multi-agent settings.