Fairness in Multi Agent Reinforcement Learning (MARL)

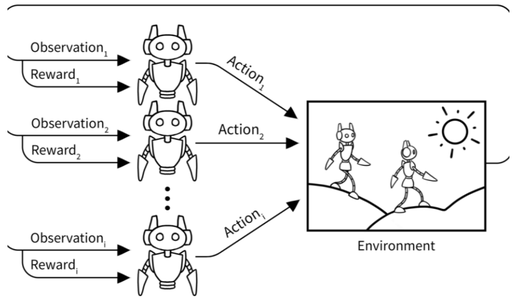
Nandini Chinta

CSE 546: Reinforcement Learning

Instructor: Prof. Dr. Alina Vereshchaka

Introduction

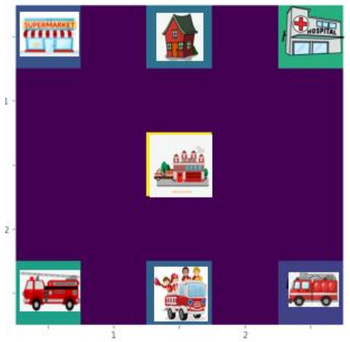
Reinforcement Learning has learned its way to solve many sequential decision-making problems. Multi Agent Reinforcement Learning (MARL) is involving multiple agents in the system. Now, proposing the fairness: equity in Multi Agent Systems to make the agents learn how to behave and cooperate in hypothetical situations. The algorithm is applied on a custom grid environment.



Problem Statement

Consider a fire station that has limited capacity of storing resources, fire engines here are agents in a town of environment and there were fires at few places in the town.

- Agents should take shortest route to reach their destination
- Since the resources are limited, agents should take resources accordingly.
- Priority should be given to fire engines going to emergency places.



This project is aimed to train the Multi Agent system to take good sequence of decisions even in hypothetical situations.

Q - Learning

This is a tabular algorithm that computes the state action (Q) value of every state in the environment by which agent knows how good to take an action at a certain state.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \overbrace{\max_a Q(s_{t+1}, a)}^{a} \right)}^{\text{learned value}}$$

Q value is estimated from above formula. Action is chosen from ε-greedy policy when the agent is training.
As we train, the agent explores more and chooses the best action for every state improving the policy.

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\ Q^*(s, a)$$

Deep Q-Network (DQN)

Deep Q-network is a Q-Learning which uses a neural network as a parameterized function approximator. DQN trains a policy which tries to maximize the discounted, cumulative reward.
Loss function is defined as:

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

- DQN tries to minimize the loss between target and the predicted to best approximate the value.
- To maintain stability in training, we use two networks to fix targets temporarily.
- Online network, to get Q value and the target network that includes all parameter updates in the training. Both networks will be synchronized after *C* updates.

Double Deep Q-Network (DDQN)

Double Deep Q-network is a double Q-Learning with two Q-networks. One to obtain best action and the other to evaluate the Q value for the action taken.

Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau << 1$
for each iteration **do**
  for each environment step **do**
    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$
    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
  for each update step **do**
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute target Q value:
      $Q^*(s_t, a_t) \approx r_t + \gamma\, Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$
    Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$
    Update target network parameters:
      $\theta' \leftarrow \tau * \theta + (1-\tau) * \theta'$

Environment

Environment here is a grid world with three agents(fire engines) trying to collect the resource at Fire Station and re-route to its destinations (Locations with fire) in shortest path and saving as many entities as possible.

Actions: Left, Right, Up, Down

Rewards:

- +1: when moving closer to the goal, -1: elsewise
- +10: when the agent has collected enough resource to put out fire.
- + #entities_saved: when reached its destination.

Scenarios applied for MARL:

Limited Resource: When agents do not have enough resource to put out fire, the agent with a public location destination will be given priority and remaining agents share the resources equally saving as many entities as possible.

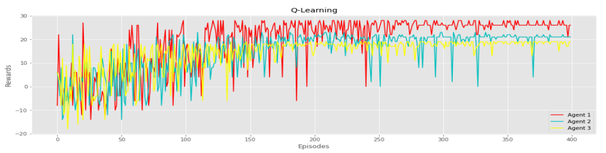
Limited Time: The algorithm is applied to train the agents to reach the goal in less than given time.

Individual Priorities: Agents with high priority will never be neglected in any case. For agents with equal priority, resources are shared equally among them and one having lesser time to reach the goal will be given priority.

Results

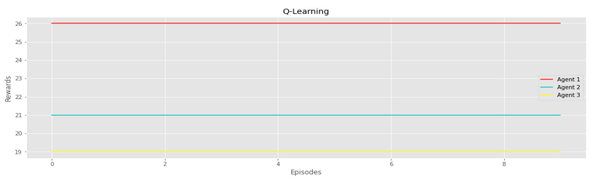
The agents are trained over 3 algorithms

Performance of 3 agents on Q-Learning

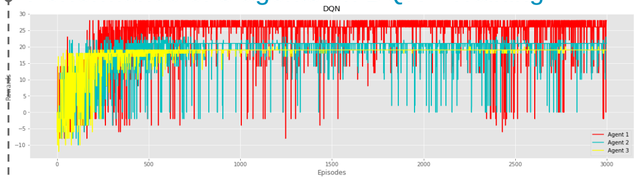


Average Rewards of 3 agents on Q-Learning

Performance of 3 agents on Q-Learning - Testing

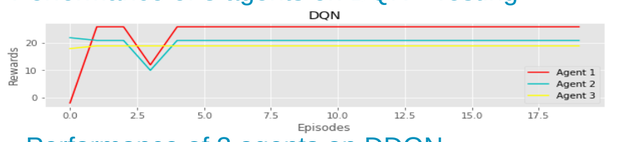


Performance of 3 agents on DQN - Training

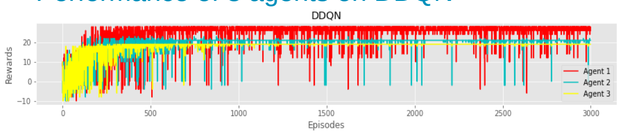


Average Rewards of 3 agents on DQN - Training

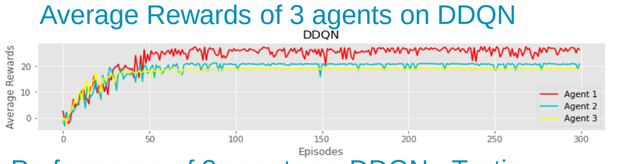
Performance of 3 agents on DQN - Testing



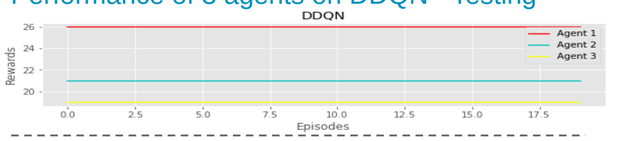
Performance of 3 agents on DDQN



Average Rewards of 3 agents on DDQN



Performance of 3 agents on DDQN - Testing



Conclusion

To conclude, Reinforcement Learning algorithms Q-Learning, DQN and DDQN have been applied on Multi Agent system with the scenarios of agents reaching their respective goals in hypothetical situations. When comparing results from all three algorithms, it is quite evident that DDQN performed well and all agents have reached their respective destinations

References

1. CSE 546 Lecture slides
2. Learning Fairness in Multi-Agent Systems
3. Stability, Fairness, and Scalability of Multi-Agent Systems
4. Reinforcement Learning as a Framework for Ethical Decision Making
5. Fairness in Multi-Agent Sequential Decision Making