Competitive Multi-Agent Reinforcement Learning

Jaishyam Ramalingam Balasubramanian Rao
Department of Computer Science
50417069
`jaishyam@buffalo.edu`

Rohit
Department of Computer Science
50412040
`vayyagar@buffalo.edu`

Karan
Department of Computer Science
50354506
`kshah23@buffalo.edu`

Muskan Arora
Department of Computer Science
50370523
`muskanar@buffalo.edu`

GitHub Repo:
`https://github.com/avsr2k/deep-learning`

Abstract

The problems of reinforcement learning involve an agent interacting with an environment. The goal of the agent is to learn to make a series of good actions which maximizes the rewards in the environment. For this project we built multi-agent hide and seek environments and used actor-critic algorithm to solve the environment.

There will be two agents in this game, the hider and the seeker, who will have opposing goals and work against each other. The hiding agent must reach safe position before being caught by the seeking agent, similarly the seeking agent must catch the hiding agent before they reach the goal. Though it seems simple, but during training MARL we run into equilibrium problem. We have ensured that both the agents converge in the environment.

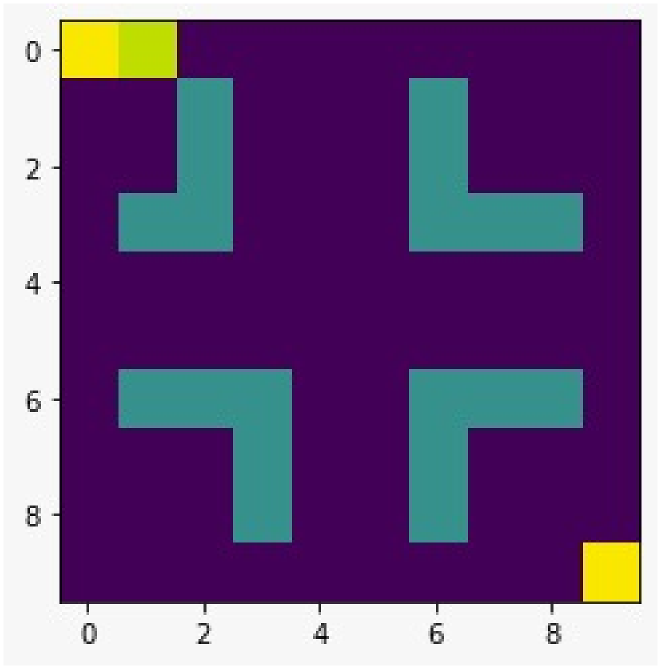
In this project, we designed a grid world environment simulating hide and seek environment and used TD actor-critic algorithm to solve the environment.

1 Environment

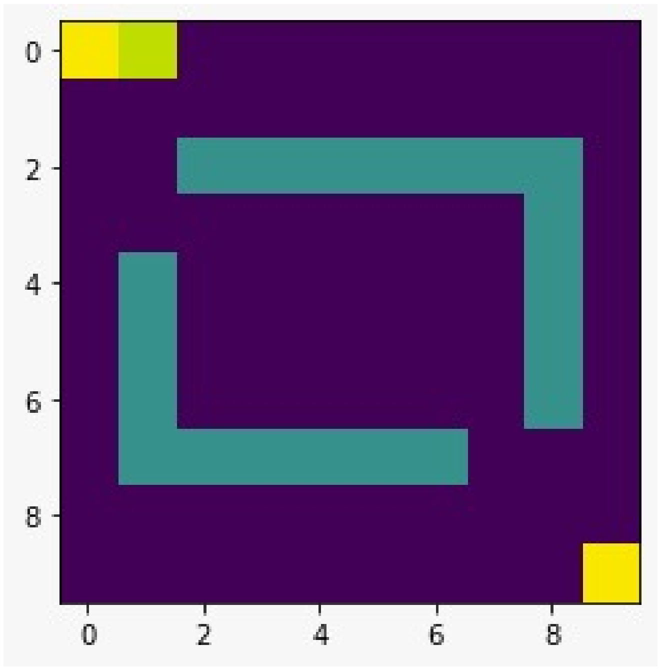
In this project we have implemented the actor-critic algorithm in two environments, which are grid-world with of size 10x10 with obstacles.

The agent which must hide is placed in the bottom right of the environment, and the goal position is at top right corner. The seeking agent is right next to the goal position making sure that no agent has any starting advantage.

The 10x10 environment with obstacles looks as follows:



The other type of obstacles looks as follows:



The importance of the obstacles is to make sure the agent learns to navigate in complex environments.

The reward for hider agent is:

- -0.5 when the agent gets further from the goal position or distance from the goal stays the same

- -0.5 when the agent gets closer to the seeker agent
- -0.1 when the agent gets closer to the goal position
- -0.1 when the agent gets further away from the seeker agent
- -1 when the agent gets caught by the seeker
- +1 when the agent gets to the goal position.

And, for seeker agent is:

- -0.5 when the agent gets further from the hider agent or distance from hider agent stays the same
- -0.1 when the agent gets closer to the hider agent
- +1 when the agent catches the hider agent.

2 Algorithm

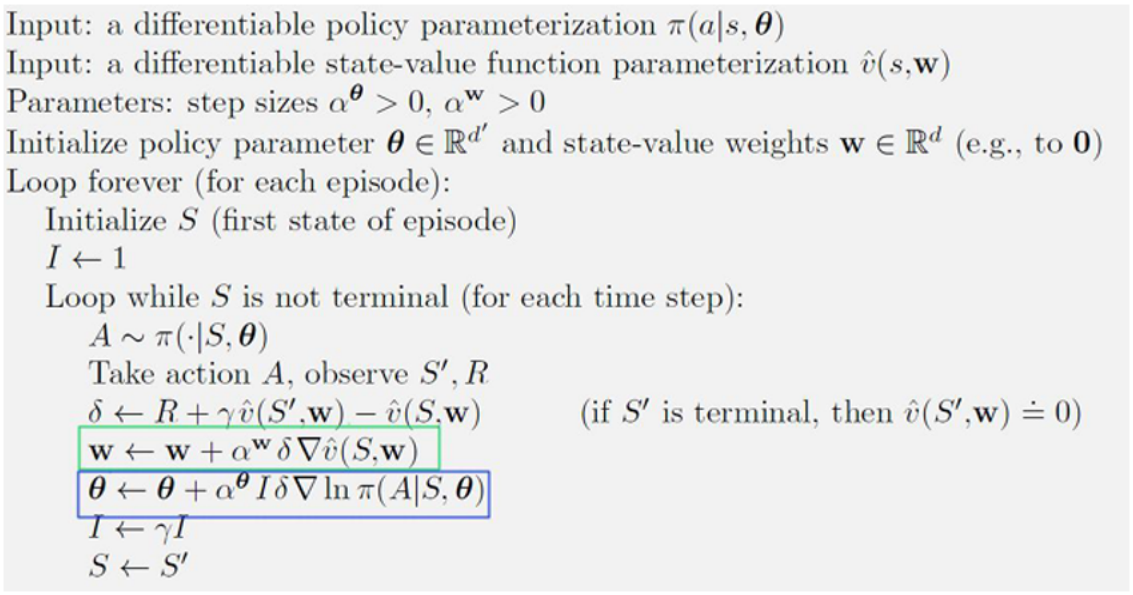
Actor-Critic is a family of algorithms which falls under the intersection of value-based algorithms and policy-based algorithms.

In actor-critic we overcome both the drawbacks of the policy-based and value-based algorithms. In actor-critic we maintain two sets of parameters namely actor and the critic. The actor is a policy-based, which is used to take an action given a state. The critic is value-based, which is used evaluate the value function of the state. This value is then used to evaluate the action taken and update accordingly.

There are different types of actor-critic, in this project we have used TD-Actor-Critic algorithm. The formula to calculate the TD error is given by:

$$\delta_{\pi_\theta} = r + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s)$$

The entire algorithm is given as:



```
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize $S$ (first state of episode)
    $I \leftarrow 1$
    Loop while $S$ is not terminal (for each time step):
        $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
        Take action $A$, observe $S', R$
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$          (if $S'$ is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\boxed{\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\,\delta\,\nabla\hat{v}(S,\mathbf{w})}$
        $\boxed{\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\,\delta\,\nabla\ln\pi(A|S,\boldsymbol{\theta})}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```

3 Axes of Multi Agent system

This is a decentralized algorithm, meaning that each agent has its own parameters and its own rewards. Each agent gets separate rewards for their action instead of shared reward.

The hider must avoid the seeker and reach the goal position. The seeker must catch the hider agent before it reaches the goal position.

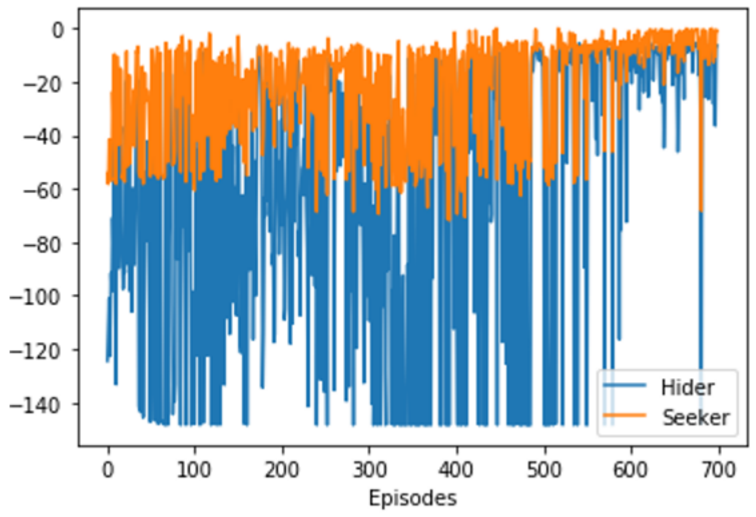
The Nash equilibrium for this environment is 50% of times the hider must succeed and other 50% the seeker must succeed.

The environment here is competitive, only one agent can win in an episode.

There are 2 agents in this environment, one is hider and other is seeker.

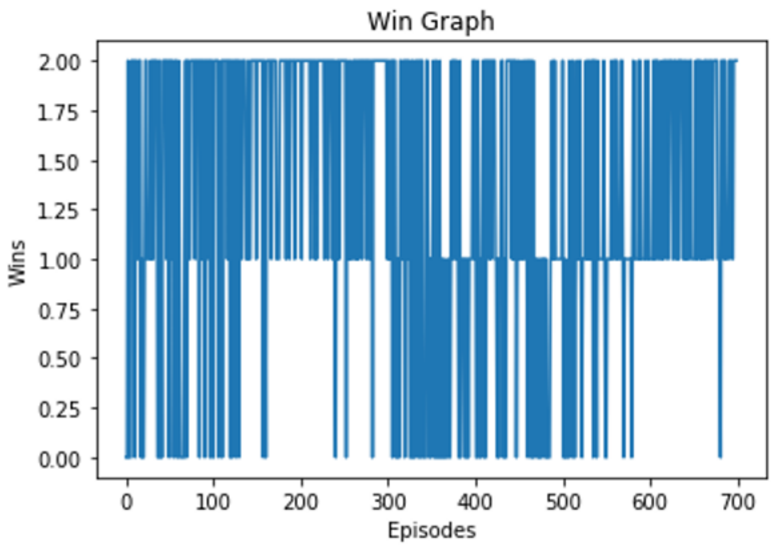
4 Training

The results of rewards for training the agent on the environment is



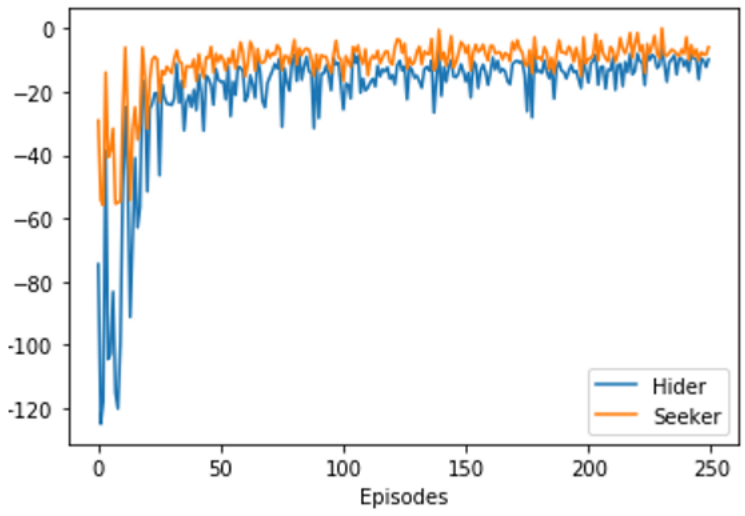
From the above graph we can see that the oscillations in the graphs are reduced towards the end. This means that that both the agents have somehow learned to learn to maximize the rewards for this environment.

The winning graph for this training is as follows



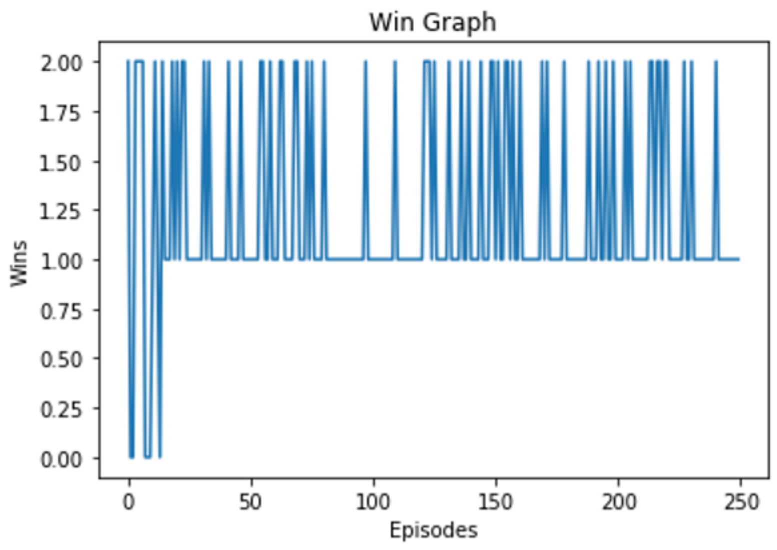
We can observe the similar oscillations in the winning graphs as seen in the rewards graph.

The results of rewards for training the agent on the other environment is



From the above graph we can see that both the agents have somehow learned to learn to maximize the rewards for this environment.

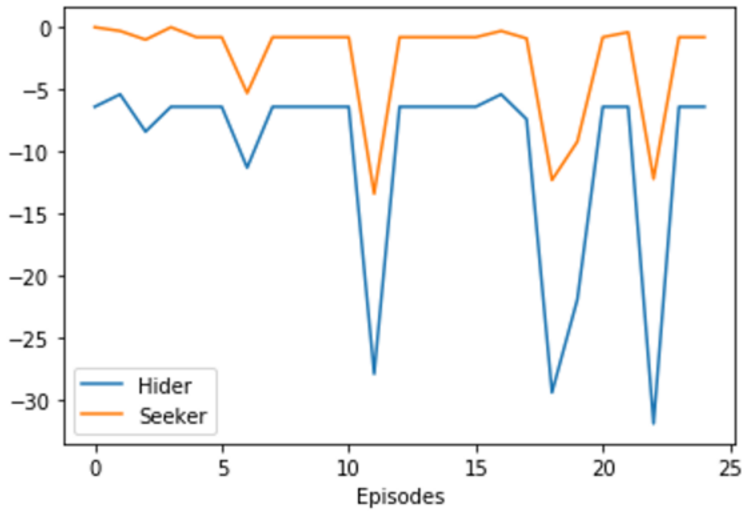
The winning graph for this training is as follows



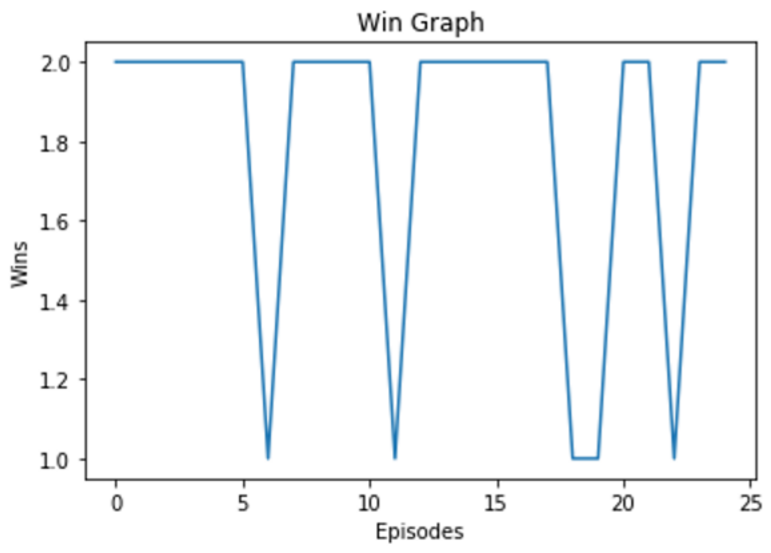
We can observe the similar pattern in the winning graphs as seen in the rewards graph.

5 Evaluation

The evaluation results after training are

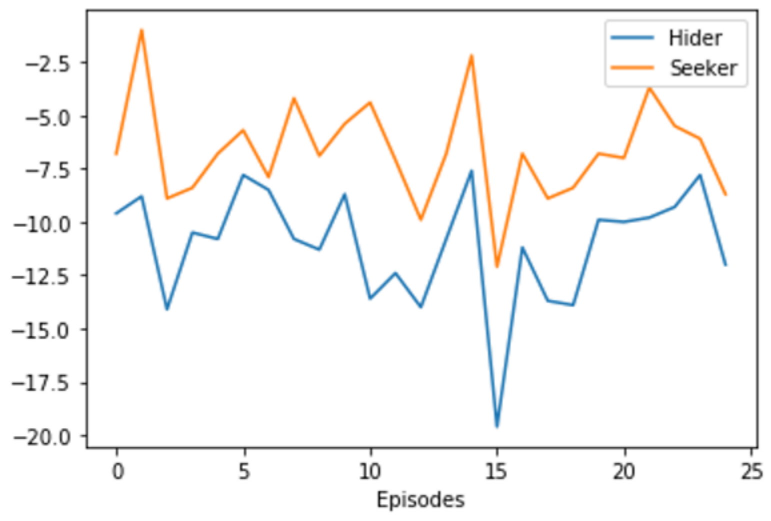


Given the complexity of the environment the oscillations in the evaluations are acceptable. The wining graph is as follows

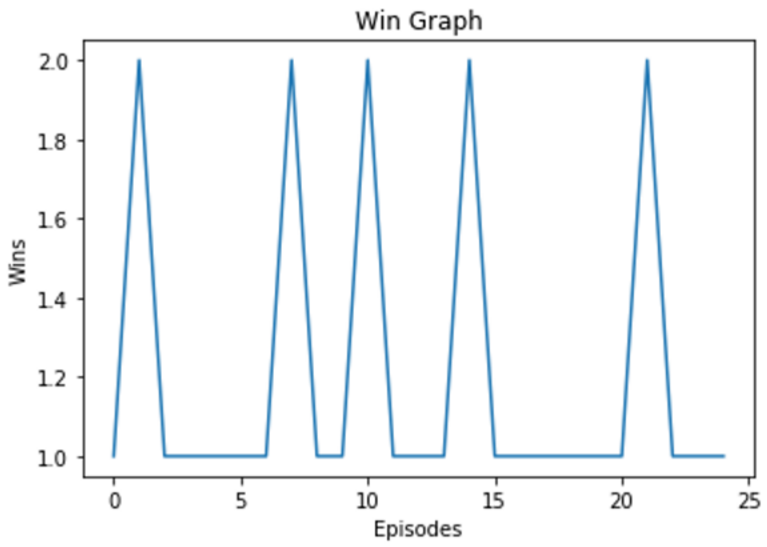


From the winning graph it is observed that the seeker agent wins more than hider agent. This could be because the agents did not fully converge, but however this environment was able to get the agent solve their own goal.

The evaluation results after training on the other environment is



Given the complexity of the environment the oscillations in the evaluations are acceptable. The wining graph is as follows:



From the winning graph it is observed that the seeker agent wins more than hider agent. This could be because the agents did not fully converge, but however this environment was able to get the agent solve their own goal.

6 Conclusion

Each environment's training never allowed just one agent to win; rather, a 50/50 split win rate was the Nash equilibrium. This demonstrated that both agents were able to adjust to their opponent's moves and understand their environment. During evaluation, both agents were able to win, however the wins favored the hider over the seeker.

References

[1]Anandkumar, A., Hsu, D., and Kakade, S. M. A method of moments for mixture models and hidden markov models. In Conference on Learning Theory, pp. 33–1, 2012.

[2]Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models. Journal of Machine Learning Research, 15:2773–2832, 2014.

[3]Azizzadenesheli, K. Reinforcement Learning in Structured and Partially Observable Environments. PhD thesis, UC Irvine, 2019.

[4]Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. Re- inforcement learning in rich-observation mdps using spec- tral methods. arXiv preprint arXiv:1611.03907, 2016. Barendregt, H. P. Introduction to lambda calculus. 1984.

[5]Bargiacchi, E., Verstraeten, T., Roijers, D., Nowe , A., and Hasselt, H. Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision prob- lems. In International conference on machine learning, pp. 482–490, 2018.

[5]Boutilier, C. Planning, learning and coordination in mul- tiagent decision processes. In Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge, TARK '96, pp. 195–210. Morgan Kaufmann Publishers Inc., 1996.

[6]Bromuri, S. A tensor factorization approach to general- ization in multi-agent reinforcement learning. In 2012 IEEE/WIC/ACM International Conferences on Web Intel- ligence and Intelligent Agent Technology, volume 2, pp. 274–281. IEEE, 2012. Bulat, A., Kossaifi, J., Tzimiropoulos, G., and Pantic, M. Incremental multi-domain learning with network latent tensor factorization. 2020.

[7]Bus oniu, L., Babus ˇka, R., and De Schutter, B. Multi-agent reinforcement learning: An overview. In Innovations in multi-agent systems and applications-1, pp. 183–221. Springer, 2010.