Asynchronous Advantage Actor Critic (A3C) Algorithm

Reinforcement Learning in Football Environment
Vimal Kumar Venugopal
vimalkumar.engr@gmail.com

***Abstract*—The google football environment is a multi-agent MDP problem that requires training the agents with an appropriate reinforcement learning technique to maximize rewards. The paper introduces to the implementation of Asynchronous Advantage Actor Critic (A3C) algorithm for training the multi-agents in the football environment. As traditional methods are non-stationary and unstable for the multi-agent environment, the A3C algorithm was chosen because: 1) it inherently supports concurrent training of multi-agent policies and 2) the addition of Advantage function helps reduce high variance during the policy gradient update; Moreover, the A3C algorithm supports for asynchronous execution which allows for effective exploration of state space and also for faster execution with parallelized hardware. After a 30 hour training on an 8 core Google VM, the two trained players were able to win 17% of the games presented to them during evaluation. The code was implemented using Ray RLLib library.**

***Index Terms*—Google Football, Multi-Agent, Actor-Critic, Asynchronous, A3C, Model**

I. Introduction

Project Goal: The goal of this paper is to configure a multi-agent Asynchronous Advantage Actor-Critic (A3C) algorithm, train it in a modified Google football multi-agent environment and evaluate the trained agent against three provided baselines, comparing their win rate % along with two metrics: mean episode reward and mean episode length.

Reason for choosing metrics: Mean episode reward and mean episode length were chosen because both are key indicators that summarize how the episode went. A higher value of mean episode reward is a result of the agents in the environment taking actions that lead to maximum rewards, with the agent scoring the goal carrying maximum reward(+1). Thus, mean episode reward can be correlated with winning. On the contrary, mean episode length metric shows how optimal the agents acted in the environment. A well-trained agent should know the optimal policy, where 'optimal' in this case is terminating an episode by scoring a goal. Given the policy is optimal, the agent should score a goal in limited number of steps. Thus a low value of mean episode length is correlated to winning.

Football Environment as a Multi-Agent MDP: The football environment consists of 3 players on each side with the opponents team and our team's goalkeeper being controlled by Game AI. The remaining two field players interacts with the football environment over discrete time steps. At each timestep t, each of the field players selects an action a_t based on its policy π. There are two policies, one for each of our team's player, that is a mapping from states s_t to actions a_t. After each player taking action, the environment returns the next state s_{t+1} and a reward r_t. This process is repeated until either a terminal state or maximum timestep count is reached. The goal of both our agents (field players) is to maximize the rewards returned from the environment.

Challenges of a Multi-Agent Environment: With more than one agent acting in the football environment, the environment is always changing and appears non-stationary from the perspective of every individual agent. This especially becomes a challenge during training, as agents constantly try to vary its policy to the varying environment, unbeknownst to the fact that there are other agents causing the environment to change. There is also the multi-agent credit assignment problem: with the environment returning a joint reward signal for actions from multiple agents, the agents are unable to distribute the reward signal among themselves, thus not learning which of their specific actions contributed to the reward.

Traditional RL methods: Traditional reinforcement learning methods such as deep neural networks (Q-learning) cannot used as the observed data encountered by an online RL agent is non-stationary. Nor the addition of experience replay[1] will help, as our environment is non-stationary due to multi-agents and renders the replay buffer unstable. Moreover, Q-learning is an off policy algorithm, that relies on data generated by older policies, worsening the multi-agent credit assignment problem. Traditional Policy gradients methods like REINFORCE cannot be used as they already suffer from high variance with the stochasticity of actions, which worsens with multi-agents.

A3C Algorithm for Football: In this paper, we attempt to use Asynchronous Advantage Actor-Critic (A3C) algorithm to come up with optimal policies for our two field players in the compact football environment. Though the policy gradients, suffers from high variance, the use of a value based Advantage function as a baseline, is an attempt to make the gradient changes smaller and thus improving stability of the policy gradient method. With 19 discrete actions, the action space for the football environment is small, which will also contribute to smaller gradient changes.The horizon (max number of time-steps allowed in an episode) of the football environment is restricted to 500, making it small, contributing to small gradients changes as well. A3C algorithm asynchronously executes multiple agents (workers) in parallel, on multiple instances of the environment. This asynchronous execution, apart for allowing efficient exploration of state space, also

provides practical benefits as no reliability on specialized hardware such as GPUs.

A3C in a multi-agent setting: Each of our two field players (agents) in the football environment maintains its own individual policy (mutli-agent policy). Each agent's policy is independent. The A3C algorithm will optimize multiple policies simultaneously from the joint reward signal. Also known as concurrent learning, the advantage of this approach is it is easier to learn heterogeneous policies where co-ordination among players is required to receive reward. This is especially true for our football environment as one field player may take on the role of a defender (preventing opponent players from reaching the ball), while the other player takes on the role of striker (taking the ball towards goal). Another reason why A3C is suited for our football environment is the number of agents is minimal. A3C is known to get unstable with large number of agents, but with our environment having only 2 agents will not be a problem. The fact that our football environment is a fully observable MDP and not a Partially Observable MDP (POMDP) helps both agents receive the actual state of the environments instead of two different partial observations which would have contributed towards increasing variance in the policy gradients.

II. Method

Policy Gradient Equation: The policy gradient equation(fig.1) is at the heart of the A3C algorithm. This model free reinforcement learning works by collecting a trajectory of one episode using current policy, and then uses it to update the policy parameter. Since the policy parameter is updated based on Monte Carlo (G_t), which is based on random samples, this leads to different trajectories during training. This causes leads to high variance, noisy gradients & instability.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta}[\nabla_\theta log\pi\theta(a_t|s_t)G_t]$$

Fig. 1. Policy Gradient Equation

Advantage Actor Critic Equation: High variance and instability of the policy gradient equation can be fixed by making the gradient changes smaller, which leads to more stable updates. This can be done subtracting the expected rewards Q(s,a) by a baseline function b(s). The baseline function b(s) is a value function(V(s) for Advantage equation. The Advantage value can be considered as how better an action (Q(s,a)) is compared to the average action (V(s)) at a state. By Bellman's equation, the policy gradient can be re-written with just the value function as shown in the last line (fig.2).

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta}[\nabla_\theta log\pi\theta(s,a)A^w(s,a)]$$
$$A(s_t,a_t) = Q_w(s_t,a_t) - V_v(s_t)$$
$$Q(s_t,a_t) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1})]$$
$$A(s_t,a_t) = r_{t+1} + \gamma V(s_{t+1}) - V_v(s_t)$$
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta}[\nabla_\theta log\pi\theta(s,a)(r_{t+1} + \gamma V(s_{t+1}) - V_v(s_t))]$$

Fig. 2. Advantage Actor Critic Equation

Actor Critic Model: The Advantage Actor Critic Equation

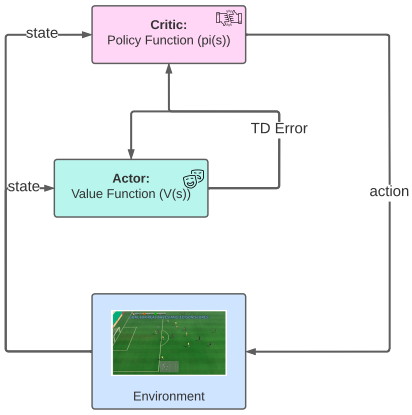

Fig. 3. Actor Critic Model

is an actor critic model in which the actor is the policy function that provides action given a state and the critic is the value function which evaluates the actions taken by the actor. The actor learns based on the updates from policy gradients. The critic computes the value function that is used to evaluate the actions produced by the actor. In our implementation, both the actor and critic are represented by one neural network with two outputs.

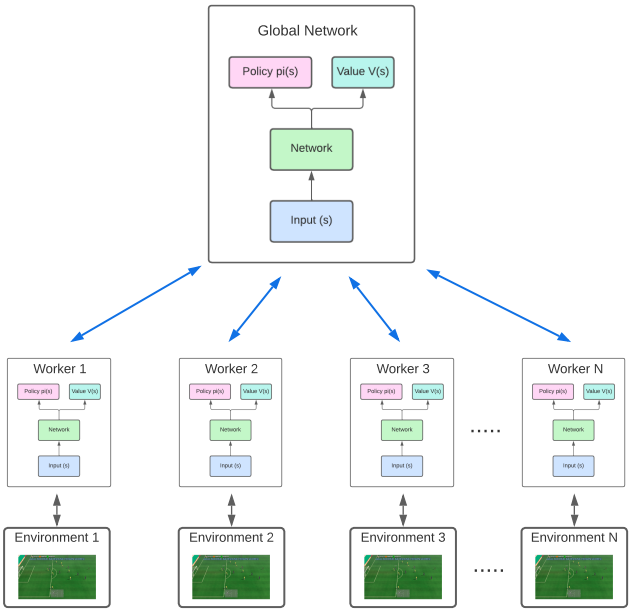

Fig. 4. Asynchronous Advantage Actor Critic (A3C) Model

Asynchronous Advantage Actor Critic Model: In A3C algorithm multiple copies of the actor critic models (workers) are executed in parallel on multiple instances of the environment with each worker maintaining its own policy π

and estimate of the value function V(s). The algorithm also maintains a global copy of the policy and value function known as global network. Every time a worker reaches terminal state or t_{max}, the worker 'asynchronously' updates the neural network weights of a global network. One advantage of this model is that multiple workers running in parallel will explorer different parts of the football environment. The second advantage is reduction in training time due to the number of parallel actor-learners. The end result is the efficient and effective exploration of the state space.

III. Algorithms

The algorithm used to implement the A3C agent is described in fig.5. The Algorithm starts by initializing the global network parameters for the global policy $\pi(a_t|s_t;\theta)$ and value function $V(s_t;\theta_v)$. Each worker starts with its neural network weights from the global network. The gradient values and the initial state is reset before the worker starts. The worker thread interacts with the environment by taking action a_t based on its own internal policy $\pi(a_t|s_t;\theta')$ at state s_t. The environment returns back a reward r_t and the next state s_{t+1}. The worker interacts with the environment until a terminal state or the max timesteps t_{max} is reached.

Asynchronous Advantage Actor Critic (A3C) Algorithm:
Init Global Network Parameters θ & θ_v
For each worker:
  Init worker specific parameters θ'=θ & θ_v'=θ_v
  Reset gradients dθ <-- 0 & $d\theta_v$ <-- 0
  Init state s_t
  Interact with Environment until t_{max} or done:
    Perform action a_t from policy π ($a_t|s_t$; θ_v') at state s_t
    Receive reward r_t and new state $s_{t=}$ s_{t+1}
  Calculate accumulated rewards R = ri + ɣR
  Calc TD Error from accumulated rewards R
  Calc value & policy loss from TD Error
  Update worker's gradients:
    dθ <-- dθ + [policy loss]
    $d\theta_v$ <-- $d\theta_v$ + [value loss]
Perform asynchrous update on Global network:
  Update θ using dθ
  Update θ_v using $d\theta_v$

Fig. 5. A3C Algorithm

Once the episode ends, the worker calculates the value loss & policy loss and updates it neural network weights through function approximation. The worker then asynchronously updates the Global network's weights from its own weight correction. The worker then resets its network and starts over again. The A3C algorithm can support as many parallel workers as the number of CPUs available. The A3C algorithm thus relies on parallel actor-learners and accumulated updates for improving training stability.

IV. Results and Analysis

Experimental Setup: Ray RLLib's implementation of A3C algorithm was executed on the football environment. The experiment was run on a Google Cloud VM with 8 CPU cores, 32 GB RAM and no GPU. Multi-agent policies were trained using A3C algorithm for a total of 50 million timesteps. For evaluation, the trained checkpoint along with the provided baselines were run for a total of 100 episodes and the test performance were analysed.

Metric 1: Fig. 6 shows the comparison of the episode reward mean metric against baselines & A3C agent. The episode reward mean was chosen as it provides a summary of how successful the episode was at each timestep. While the baselines agents was able to achieve mean reward values of over 1.0 at 50M timesteps, the A3C agent was only able to achieve a mean reward of about 0.65 at end of the training. The reason for this sub-optimal performance being the inefficient sampling of the A3C algorithm on the football environment. Given that the accuracy of policy gradients depends on optimal value function V(s), the A3C algorithm needs to visit each state-action pair many number of times. The football environment is non-stationary due to the environmental dynamics: ball position, opponent position etc. Along with 19 different action spaces, it makes the number of state-action pairs exponential, causing the number of samples required to be visited for an optimal value function exponential as well. Also notice from fig. 6 that the slope of the A3C agent is slow and gradual, unlike the baseline agents which grow to max rewards at a much steeper angle. The slower convergence is due to the sample inefficiency as described above along with the high variance problem of policy gradients. The low episode reward mean indicates that the A3C agent will not yield high win rate when compared to the baselines as the low episode reward correlates to the agent not having learnt the policy that lead to a successful goals (max reward=+2) yet.

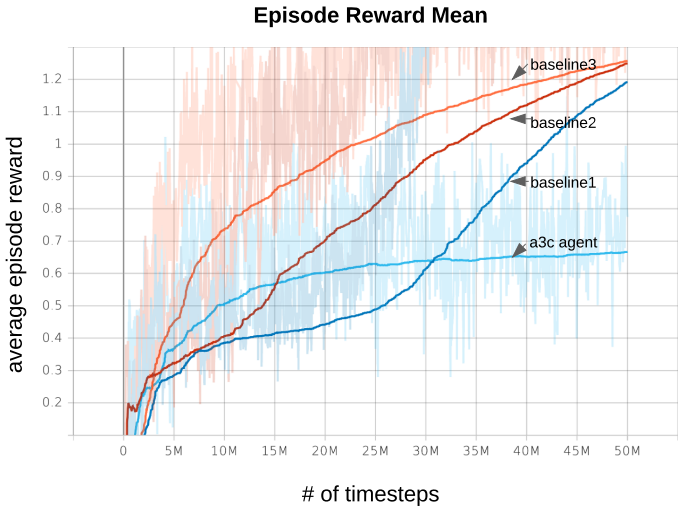

Fig. 6. Metric#1 - Episode Reward Mean

Metric 2: Fig.7 shows the comparison of the mean episode length of the different agents. The mean episode length was chosen as a metric to judge the win rate as it is summarizes how long the episode lasted for. At the start of learning, the agent will attempt multiple actions, most of them causing no

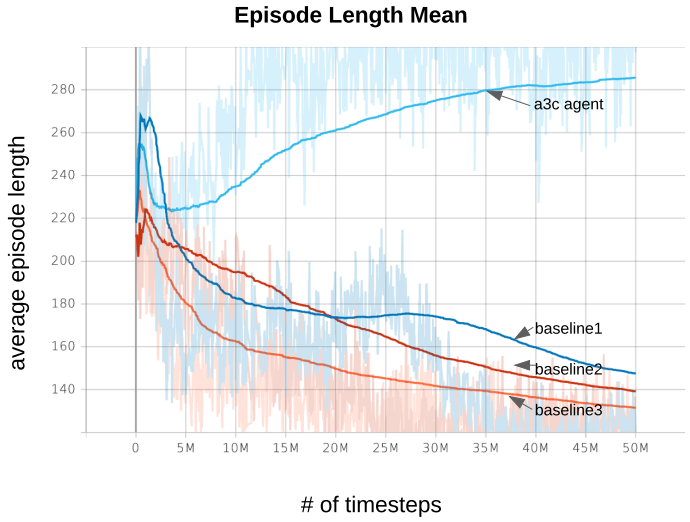

Fig. 7. Metric#2 - Episode Length Mean

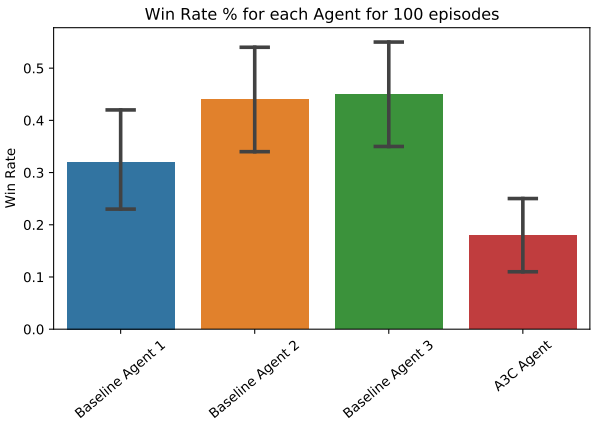

Fig. 8. Agent Evaluation - Win Rate

rewards from the environment yet taking up timesteps. As the agent learns the environment, the actions taken are more concise and reward oriented. When the policy gets closer to the optimum, the agent should know exactly what action to take at each state so as to maximize rewards. Thus a low number of steps in an episode could be translated to the agent acting optimally to score goals (maximum reward). Thus the knowledge of the mean episode length is a metric to decide if the agent will yield a high win-rate or not. Since lower the value is better, fig. 7 reveals that baseline agents 2 & 3 may yield a better win rate than the A3C agent, which has the highest mean episode length of all. The figure also shows there is a positive slope of the A3C agent while the baseline agents have negative slopes. The reason for this being, A3C agent being an on-policy algorithm, we are sampling directly from the policy we are optimizing. Learning on A3C agent occurs by sampling multiple trajectories from the existing policy. The more state-action pairs visited, the more A3C agent will learn. This is unlike PPO methods which stochastically select the optimal action from multiple policies. Thus taking more episode timesteps correlates to a low win-rate performance against the opponent football team.

Win rates: Looking at the win rates (fig. 8), the A3C agent performed sub-optimally compared to the baseline agents winning only about 18 of the 100 episodes presented to it. As from the metrics analysis, the reason for A3C agent's sub-optimal performance could be attributed to the comparatively low average reward collected during the episodes in training. The other reason being not taking optimal actions and hence prolonging the length of the episode during training. Both of these factors cause the A3C agent to behave less optimally. The converse is true for baseline 3 agent which scores a high average episode taking minimal number of steps per episode during training and thus performs the best of all agents, scoring a 45% win rate. This yet again proves that our chosen metrics are correlated to the win rate.

V. Pitfalls & Problems

Although A3C algorithm was able to concurrently train our two field player agents with their independent policies in the football environment, the win-rate performance was sub-optimal compared to the baselines. This could be due to the following reasons.

Sample Complexity: Being a model free algorithm, there is no priori model and A3C starts off with a uniform distributions over all actions at a given state. As training proceeds, the algorithms forms multi-model distributions of action spaces over the states. However, to determine the optimal action for a state, the algorithm has to observe the outcomes of many different state-action pairs multiple times. Moreover, since our football environment has two agent, the joint action space of the agent doubles. These factors exponentially increases the sample complexity in our dynamic football environment.

Slow Convergence: The high sample complexity translates to slower convergence as the A3C algorithm is trying to optimize the policy after each run. Though having asynchronous update through the multiple workers helps, we are limited by the physical hardware. In our experiment, the 8 core CPU meant only a limited number of asynchronous updates could be done in parallel. Moreover, the gradual positive slope of the average episode reward during training (fig. 6) indicates our agent has not optimized yet and could benefit from more timesteps.

High variance: Being an on-policy algorithm, A3C has to sample from the policy the algorithm is trying to optimize. As football environment is stochastic, the policy gradient updates obtained from observing a stochastic trajectory of a policy will end up having high variance. Although Advantage function helps reduce variance, the algorithm inherently will still have high variance from the environment.

Multi-agents: Although A3C algorithm allows for concurrent training of multi-agent policies, having more then one agent in the environment, makes the observations seen by the other agent stochastic and not congruent to the agent's past experience of a particular state-action pair. The existence of multiple agents adds noise to the events during training.

Accumulating noise over multiple timesteps results in high variance problem.

Muti-agent credit assignment: In the football environment, actions from both agents generate only a global reward. This makes it difficult for each agent to decide which of their actions contributed for the reward from the environment. This causes learning to be affected as an agent's policy may be incorrectly optimized based on the other player's actions. Without the knowledge of their contributions, it is not possible for multi-agents to collaborate or co-operate to achieve the common reward.

VI. Conclusion

Although the A3C algorithm was able to train the two field agents, the performance was not close to the provided baseline agents. Given more time, the winrate performance of our implemented agent could have been improved by including reward shaping. Collaborative and competitive behavior could have been obtained from the agents with appropriate choice of reward structure. For instance, while one player is dribbling, the other player could have been incentivized for keeping the other players out of the way or being at a location closer to the goal to receive a pass.

The actor critic method could have also been extended by augmenting the critic (value V(s)) function with additional information about the policy of the other field player[6]. This knowledge about the other player's policy could lead to collaboration and co-operation. The actor (policy $\pi(s)$ function can then act independently with access to only the local information received by its own player agent. This framework of centralized training with decentralized execution could have allowed for improved winrate by our multi-player RL agent in the football environment.

In addition to the centralized critic and decentralized action framework, the problem of multi-agent credit assignment could be solved by using a counterfactual baseline[7] in which the centralized critic computes an agent specific advantage function that compares the estimated return for the current joint action to a counterfactual baseline that marginalises out a single agent's action, while keeping the other agents' action fixed.

Moreover, the critic representation could also be updated from A3C implementation to allow for the counterfactual baseline to be computed efficiently. Instead of calculating the value function $V(s)$ at each state, the critic can compute the Q-values $Q(s, a)$ for all the different actions of a given agent, conditioned on the actions of all the other agents[7]. Since a single centralised critic is used for all agents, all the Q-values for all agents can be computed in a single batched forward pass.

This paper, thus implements an A3C reinforcement learning algorithm to train the multi-agents in the google football environment and compares the win rate against the provided baseline agents. This paper also chose two metrics and shows how the RL agents performance on these metrics can be correlated to the winrate.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

[2] Mnih V., Badia A., Mirza M., Graves A., Lillicrap T., Harley T., Silver D., and Kavukcuoglu K., Asynchronous methods for deep reinforcement learning International Conference on Machine Learning , page 1928–1937. (2016)

[3] Sutton, R.S. & Barto, A.G., 2018. Reinforcement learning: An introduction, MIT press.

[4] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016). https://doi.org/10.1038/nature16961

[5] Gupta, Jayesh K. et al. "Cooperative Multi-agent Control Using Deep Reinforcement Learning." AAMAS Workshops (2017).

[6] Lowe, Ryan et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments." ArXiv abs/1706.02275 (2017): n. pag.

[7] Jakob N. Foerster et al. "Counterfactual Multi-Agent Policy Gradients". In: CoRR abs/1705.08926 (2017). arXiv: 1705.08926. url: http://arxiv.org/abs/1705.08926.

[8] Ray RLLib A3C Algorithm, https://docs.ray.io/en/latest/rllib/rllib-algorithms.html#a3c

[9] Medium: Implement A3C algorithm to play breakout, https://medium.com/@shagunm1210/implementing-the-a3c-algorithm-to-train-an-agent-to-play-breakout-c0b5ce3b3405

[10] Medium: Comparing actor-critic methods, https://jonathan-hui.medium.com/rl-actor-critic-methods-a3c-gae-ddpg-q-prop-e1c41f268541