Quiz 11

Algorithm	Value Based/Policy Gradient/AC
DQN	Value Based
Double DQN	Value Based
Dueling DQN	Value Based
Reinforce	Policy Gradient
A2C	AC
A3C	AC

Algorithm	Objective Function and Parameter Updates
DQN	Objective Functions:- $$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$ Parameter Updates:- $$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$
Double DQN	Objective Functions:- $$Q^*(s_t, a_t) \approx r_t + \gamma\, Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$$ Parameter Updates:- Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$ Update target network parameters: $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$
Dueling DQN	$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in \lvert A \rvert} A(s, a'; \theta, \alpha)\right)$$ $$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{\lvert A \rvert} \sum_{a'} A(s, a'; \theta, \alpha)\right)$$ Parameter Updates:-

	Here, θ denotes the parameters of the convolutional layers, while α and β are the parameters of the two streams of fully-connected layers. $$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha)$$
Reinforce	Objective Function:- $$J_{avR}(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(s,a) R_s^a$$ Parameter Updates:- $$\nabla_\theta J(\theta) = \mathbb{E}_\pi[Q^\pi(s,a) \nabla_\theta \ln \pi_\theta(a\|s)]$$
A2C	Objective Function and Parameter Updates :- ■ For the true value function $V_{\pi_\theta}(s)$, the TD error δ_{π_θ} $$\delta_{\pi_\theta} = r + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s)$$ ■ is an unbiased estimate of the advantage function $$\begin{aligned}\mathbb{E}_{\pi_\theta}[\delta_{\pi_\theta}\|s,a] &= \mathbb{E}_{\pi_\theta}\left[r + \gamma V_{\pi_\theta}(s')\|s,a\right] - V_{\pi_\theta}(s) \\ &= Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s,a)\end{aligned}$$ ■ So we can use the TD error to compute the policy gradient $$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta_{\pi_\theta}]$$ ■ In practice we can use an approximate TD error, that requires one set of parameters w $$\delta_w = r + \gamma V_w(s') - V_w(s)$$
A3C	Objective Function and Parameter Updates :- ■ For the true value function $V_{\pi_\theta}(s)$, the TD error δ_{π_θ} $$\delta_{\pi_\theta} = r + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s)$$ ■ is an unbiased estimate of the advantage function $$\begin{aligned}\mathbb{E}_{\pi_\theta}[\delta_{\pi_\theta}\|s,a] &= \mathbb{E}_{\pi_\theta}\left[r + \gamma V_{\pi_\theta}(s')\|s,a\right] - V_{\pi_\theta}(s) \\ &= Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s,a)\end{aligned}$$ ■ So we can use the TD error to compute the policy gradient $$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta_{\pi_\theta}]$$ ■ In practice we can use an approximate TD error, that requires one set of parameters w $$\delta_w = r + \gamma V_w(s') - V_w(s)$$

Key Features

1.) DQN

a. Use experience replay:
    i. Break correlations in data, bring us back to iid setting.
    ii. Learn from all past policies
b. Freeze target Q-network
    i. Avoid oscillations
    ii. Break correlations between Q-network and target
c. Clip rewards or normalize network adaptive to sensible range
    i. Robust gradients

2.) Double DQN

a. Double Q-learning can be used at scale to successfully reduce this overoptimism, resulting in more stable and reliable learning.
b. Double DQN uses the existing architecture and deep neural network of the DQN algorithm without requiring additional networks or parameters

3.) Dueling DQN

a. Intuitively, the dueling architecture can learn which states are (or are not) valuable, without having to learn the effect of each action for each state.
b. The dueling architecture represents both the value V(s) and advantage A(s, a) functions with a single deep model whose output combines the two to produce a state-action value Q(s, a).

4.) Reinforce

a. Policy gradient algorithms are widely used in reinforcement learning problems with continuous action spaces.
b. The basic idea is to represent the policy by a parametric probability distribution $\pi\theta(a|s) = P[a|s; \theta]$ that stochastically selects action a in state s according to parameter vector θ.

5.) A2C

a. Model contains value and policy gradient.
b. Actor does policy gradient and critic is used to train the actor by TD error.
c. Actor performs action and critic guides the actor on how good is the taken action.

6.) A3C

a. Uses asynchronous threads for training.
b. It uses N copies of agents working parallel on the environment, collecting samples and computing gradients.

Algorithm	**Pros**	**Cons**
DQN	1.) Uses Neural Networks for estimation 2.) Highly useful when there are innumerous states.	1.) DQN is overoptimistic few times.

Double DQN	1.) Unlike DQN, Double DQN is not overoptimistic. 2.) Double DQN uses the existing architecture and deep neural network of the DQN algorithm without requiring additional networks or parameters 2.) It uses two NN for estimation, hence gives better results.	1.) Need to maintain 2 neural networks.
Dueling DQN	1.) It can learn which states are valuable, without learning the effect of each action every time.	
Reinforce	1.) Highly useful in continuous action space 2.) Reinforce is generally unbiased.	1.) Gives a lot of variance.
A2C	1.) Has less variance. 2.) Takes the best out of both Value Based and Policy Gradient.	1.) has two function approximators to handle.
A3C	1.) No Experience Replay Buffer needed 2.) Even one can explicitly use different exploration policies in each actor-learner to maximize diversity.	1.) Complex environment to build. 2.) need to properly handle the threads.