UNIVERSITY OF LIVERPOOL

Generative Adversarial Networks (GANs) and Deep Reinforcement Learning

Deep Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning algorithm that falls somewhere between supervised and unsupervised. It cannot be classified as supervised learning because it doesn't rely solely on a set of labelled training data, but it's also not unsupervised learning because we're looking for our reinforcement learning agent to maximise a reward.

To understand Reinforcement Learning better, imagine that you want your computer to play chess with you. The first question to ask is this: Would it be possible if the machine was trained in a supervised fashion? In theory, yes. But there are two drawbacks that you need to consider. Firstly, to move forward with supervised learning you need a relevant dataset. Secondly, if we are training the machine to replicate human behaviour in the game of chess, the machine would never be better than the human, because it's simply replicating the same behaviour. So, by definition, we cannot use supervised learning to train the machine. But is there a way to have an agent play a game entirely by itself? Yes, that's where Reinforcement Learning comes into play. Reinforcement Learning is a type of machine learning algorithm that learns to solve a multi-level problem by trial and error. The machine is trained on real-life scenarios to make a sequence of decisions. It receives either rewards or penalties for the actions it performs. Its goal is to maximise the total reward. By Deep Reinforcement Learning we mean multiple layers of Artificial Neural Networks that are present in the architecture to replicate the working of a human brain.

In reinforcement learning, we have a few key components to be discussed: 1) Agent: Agent (A) takes actions that affect the environment. Citing an example, the machine learning to play chess is the agent. 2) Action - It is the set of all possible operations/moves the agent can make. The agent decides on which action to take from a set of discrete actions. 3) Environment - All actions that the reinforcement learning agent makes directly affect the environment. Here, the board of chess is the environment. The environment takes the agent's present state and action as information and returns the reward to the agent with a new state. For example, the move made by the bot will either have a negative/positive effect on the whole game and the

arrangement of the board. This will decide the next action and state of the board. 4) State - A state (S) is a particular situation in which the agent finds itself. 5) Reward (R) - The environment gives feedback by which we determine the validity of the agent's actions in each state. It is crucial in the scenario of Reinforcement Learning where we want the machine to learn all by itself and the only critic that would help it in learning is the feedback/reward it receives. For example, in a chess game scenario it happens when the bot takes the place of an opponent's piece and later captures it.

Reinforcement Learning algorithms can be divided into two types – model-based and model-free. Model-based Reinforcement Learning algorithms have an agent trying to comprehend its environment and build a model for it based on its relations with this environment. In such a system, selections take priority over the outcomes of the actions i.e., the greedy agent will always try to perform an action that will get the maximum reward irrespective of what that action may cause. On the other hand, model-free algorithms seek to learn the consequences of their actions through experience via algorithms such as Policy Gradient, Q-Learning, etc. In other words, such an algorithm will carry out an action multiple times and will adjust the policy (the strategy behind its actions) for optimal rewards, based on the outcomes. Think of it this way, if the agent can predict the reward for some action before actually performing it thereby planning what it should do, the algorithm is model based. While if it actually needs to carry out the action to see what happens and learn from it, it is model-free. This results in different applications for these two classes, for e.g., a model-based approach may be the perfect fit for playing chess or for a robotic arm in the assembly line of a product, where the environment is static and getting the task done most efficiently is our main concern. However, in the case of real-world applications such as self-driving cars, a model-based approach might prompt the car to run over a pedestrian to reach its destination in less time (maximum reward), but a model-free approach would make the car wait till the road is clear (optimal way out).

Reinforcement Learning involves managing state-action pairs and keeping a track of the value (reward) attached to an action to determine the optimum policy. This method of maintaining a state-action-value table is not possible in real-life scenarios when there are a larger number of possibilities. In Deep Reinforcement Learning, instead of utilising a table, we can make use of Neural Networks to predict values for actions in each state.