Reinforcement Learning: The Business Use Case, Part 1

Aishwarya Srinivasan Following
Jul 24, 2018 · 5 min read

The whirl of reinforcement learning started with the advent of AlphaGo by DeepMind, the AI system built to play the game Go. Since then, various companies have invested a great deal of time, energy, and research, and today reinforcement learning is one of the hot topics within Deep Learning. That said, most businesses are struggling to find use cases for reinforcement learning or ways to encompass it within their business logic. That shouldn't surprise us. So far, it's been studied only in risk-free, observed, environments that are easy to simulate, which means that industries like finance, health, insurance, tech-consultancies are reluctant to risk their own money to explore its applications. What's more, the aspect of "risk factoring" within reinforcement learning puts a high strain on systems. Andrew Ng, the co-chair and cofounder of Coursera has said that "reinforcement learning is a type of machine learning whose hunger for data is even greater than supervised learning. It is really difficult to get enough data for reinforcement learning algorithms. There's more work to be done to translate this to businesses and practice."

With this somewhat pessimistic view in mind, let's use Part 1 of this blog to dig a little deeper into the technical aspect of reinforcement learning. In Part 2, we'll look at some possible applications in business. At base, RL is a complex algorithm for mapping observed entities and measures into some set of actions, while optimizing for a long-term or short-term reward. The RL agent interacts with the environment and tries to learn policies, which are sequences of decisions or actionstoward obtaining the reward. In fact, RL considers immediate rewards and delayed rewards as it drives its interactions with the agent.

A Reinforcement learning model consists of an agent which infers an action which then acts on the environment to make a change, and the significance of the action is reflected using a reward function. This reward is optimized for the agent and the feedback is passed to the agent so that it can assess the next best action to take. The system learns from the previous action by recalling the best action to take in similar circumstances.

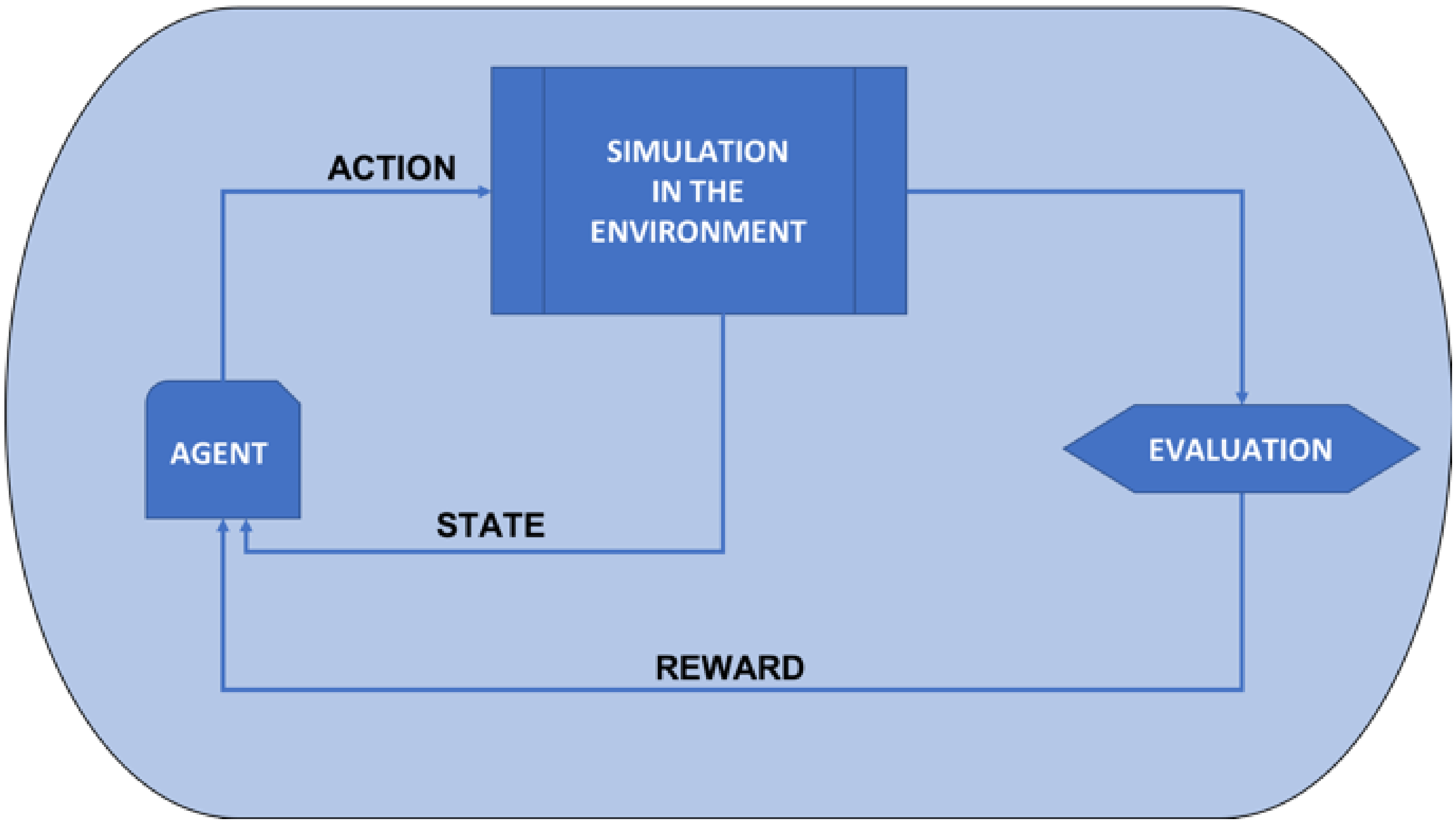

Fig 1: The Reinforcement Learning Model

From a mathematics perspective, we can look at reinforcement learning as a state model, specifically a fully observable Markov Decision process(MDP). To understand the probabilistic theory behind the MDP, we need to know the Markov property:

> "The future is independent of the past given the present"

The Markov Property is used in situations where the probabilities of different outcomes are not dependent on past states; therefore, it requires only the current state. Some people use the term "memoryless" to describe the property. In situations where you need previous states to inform the outcome, a Markov Decision Process won't work.

The environment of the model is a stochastic finite state machine, with take actions given out from the agent as inputs and where the rewards/feedback sent from environment to the agent are outputs. The overall reward function consists of immediate rewards and delayed rewards. The immediate reward is the quantitative impact of the action on the state environment. The delayed reward is the effect of the action on future states of the environment. The delayed reward is accounted for using the *'discount factor (γ)'* parameter, $0 < \gamma < 1$. A higher value for the discount factor tips the system toward far-sighted rewards, while a lower value tips the system toward immediate rewards. *X(t)* is the representation of the environment state at time '*t*'. *A(t)* is action taken by the agent at time *'t'*.

· State transition function: From one state to another in the environment as a result of the actions given out by the agent.

$$P(X(t)|X(t-1), A(t))$$

The agent is also modeled as a stochastic finite state machine, where the rewards sent from the environment is the inputs and the actions sent to the environment for the next time step is the output. *S(t)*is the current state of the agent at time *'t'* after receiving the feedback from the environment applied from the environment at *'t-1* after the action *A(t)*is prepresents the policy being built using model learning by overall reward optimization.

$$E(R(t)|X(t), A(t))$$

· State transition function: From one state to another in the agent as a result of the rewards given out by the environment.

$$S(t) = f(S(t-1), Y(t), R(t), A(t))$$

· Policy function: The policy/output function from the agent to give out the action based on the optimization of the reward function.

$$A(t) = \ \pi(S(t))$$

The goal of the agent is to find the policy P(pi), which maximizes the overall expected reward with the discount factor.

$$E[R(0) + \gamma R(1) + \ \gamma^2 R(2) + \cdots] = E\left[\sum_{t=0} R(t)\gamma^t\right]$$

The agent trained with the MDP tries to get the most expected sum of rewards from the present state. Hence, the optimal value function needs to be obtained. The Bellman equation is used for the value function, decomposed into present reward and the discounted value of the next state value.

$$v(s) = \ E[R(t+1) + \gamma\ v(S(t+1))\ |\ S(t) = s]$$

I hope you got a view of the technical aspects of Reinforcement Learning by now!!

In the next part of this post-series, we'll look at a real-world application as a business use-case of financial industry, which would be stock trading.

Keep Learning Deep!

Machine Learning Reinforcement Learning Deep Learning Artificial Intelligence Research