A QUICK INTRODUCTION TO REINFORCEMENT LEARNING

A PREPRINT

Krishna N Agaram

December 30, 2022

1 Chapter 1: What is Reinforcement Learning?

How do we learn a lot of stuff we were never taught? Well, from experience. We have tried a lot of ways and seen what happened (the environment around us told us what happened), and so by now have a good idea of what to do in which situation. This is essentially the idea of Reinforcement learning.

- The goal is for an *agent* to learn a **policy** - roughly a map from a state of the environmant to actions to be taken when in those states. That is, a policy defines the learning agent's way of behaving at a given time. The policy may be stochastic.
- A **reward signal** defines the goal in a reinforcement learning problem. On each time step, the environment sends to the reinforcement learning agent a single number, a reward. The agent's sole objective is to maximize the total reward it receives over the long run. The reward signal thus defines what are the good and bad events for the agent. Just like the way we learn, the reward signal is the primary basis for altering the policy.
- Roughly speaking, the **value** of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the *long-term* desirability of states *after taking into account* the states that are likely to follow, and the rewards available in those states. To make a human analogy, rewards are somewhat like pleasure (if high) and pain (if low), whereas values correspond to a more refined and *farsighted* judgment of how pleased or displeased we are that our environment is in a particular state. Action choices are made based on value judgments. We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run.

  Unfortunately, it is much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of obser- vations an agent makes over its entire lifetime. In fact, the most important component of almost all reinforcement learning algorithms we consider is a method for efficiently estimating values.
- Sometimes a fourth element (the first three being the policy, reward and value) of reinforcement learning is used, called the *model* (of the environment). This is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave. For example, given a state and action, the model might predict the resultant next state and next reward.

2 An Example: Tic-Tac-Toe

It is well-known that an optimal player can never lose a game of TTT. Suppose then that we had an agent trying to beat an imperfect player. The agent treats both draws and losses as a reward of 0 and a win as a reward of 1. The agent seeks to maximize its score, that is, win. Initially, all states s that are neither winning nor losing state are given a value of 0.5 (the agent initially assumes that from state s, it is equally likely to

win or lose). Winning states are valued at 1, and losing 0. We now begin our journey. While we are playing, we change the values of the states in which we find ourselves during the game. We attempt to make them more accurate estimates of the probabilities of winning. To do this, we "back up" the value of the state after each greedy move to the state before the move. More precisely, the current value of the earlier state is adjusted to be closer to the value of the later state. This can be done by moving the earlier state's value a fraction of the way toward the value of the later state. The update to the (estimated) value of s, denoted $V(s)$, can be written

$$V(s) \leftarrow V(s) + \alpha[V(s') - V(s)]$$

where α is a small positive fraction called the **step-size parameter**, which influences the rate of learning. This update rule is an example of a *temporal-difference* learning method, so called because its changes are based on a difference, $V(s') - V(s)$, between estimates at two different times.

Moves (actions) are made according to the following rule: In state s, we choose a such that $V(s \circ a)$ is maximum (here $s \circ a = s'$ is the state we land in after taking action a from s). Note that it is usually fruitful to explore another action with some small probability so that we do not restrict our search for a better solution.

Finally, after some practice, the tic-tac-toe player is able to "look ahead" and know which move to take from where and eventually wins when it can.

3 Chapter 2: Multi-arm Bandits

Reinforcement Learning uses training information that helps evaluate the actions taken rather than instructing by giving correct actions. This is what creates the need for active exploration, for an explicit trial and error search for good behavior. Purely evaluative feedback indicates how good the action taken is, but not whether it is the best or the worst action possible. Thus, we must balance the tradeoff between **exploitation** and **exploration** - exploiting the best moves at every stage greedily does not give us a chance to explore other (possibly better) moves that may lead to a higher *total* reward.

To explain and show why exploration might be better, as well as to look at a first analysis of Reinforcement Learning, we consider the simple problem of n-armed Bandits:

Definition (The n-armed bandit problem)**.** *You are faced repeatedly with a choice among n different options, or actions. After each choice you receive a numerical reward chosen from a stationary probability distribution that depends on the action you selected. Your objective is to maximize the expected total reward over some time period, for example, over 1000 action selections, or time steps.*

In our n-armed bandit problem, each action has an expected or mean reward given that that action is selected; let us call this the *value* of that action. If you knew the value of each action, then it would be trivial to solve the n-armed bandit problem: you would always select the action with highest value, and that would sum up to the maximum expected total reward. We henceforth assume that you do not know the action values with certainty, although you may have estimates.

If you maintain estimates of the action values, then at any time step there is at least one action whose estimated value is greatest. We call this a greedy action. If you select a greedy action, we say that you are exploiting your current knowledge of the values of the actions. If instead you select one of the nongreedy actions, then we say you are exploring, because this enables you to improve your estimate of the nongreedy action's value. Exploitation is the right thing to do to maximize the expected reward on the one step, but exploration may produce the greater total reward in the long run.