The Reinforcement Learning Problem

Computational approach to learning from interaction. Rather than directly theorizing about how people or animals learn, we explore idealized learning situations and evaluate the effectiveness of various learning methods. That is, we adopt the perspective of an artificial intelligence researcher or engineer. The approach we explore, called **reinforcement learning**, is much more focues on goal-directed learning from interaction than other approaches to machine learning.

Reinforcement Learning

Reinforcement learning problems involve learning what to do - **how to map situations to actions** - so as to maximize a numerical reward signal. In an essential way they are $closed - loop$ **problems because the learning system's actions influence its later inputs.**

Moreoever, the learning is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out.

In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and , through that , all subsequent rewards.

These three characteristics -

- Being closed loop
- not having direct instructions as to what actions to take
- where the consequences of actions ,including reward signals play out over extended time periods.

are the most important distinguishing features of reinforcement learning problems.

The basic idea is to simply capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal. **Clearly, such an agent must be able to sense the state of the environment to some extent and must be able to take actions that affect the state.**

The agent also must have a goal or goals relating to the state of the environment.

The formulation is intended to include just three aspects - **sensation, action and goal** .

One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off between **exploration and exploitation**.

To obtain a lot of reward, a reinforcement learning agent must prefer actions that is has tried in the past and found to be effective in producing reward.

But to discover such actions, it has to try actions that it has not selected before.

The agent has to $exploit$ what it already knows in other to obtain reward, but it also has to $explore$ in orhder to make better action selections in the future.

The agents must try a variety of actions and progressively favor those that apprear to be best.

Another key feature of reinforcement learning is that is explicitly considers the $whole$ problem of a goal-directed agent interacting with an uncertain environment, in contrast with many approaches that considers subproblems without addresing how they might fit into a larger picture.

All reinforcement learnings agents have explicit goals , can sense aspects of their environments , and can choose actions to influence their environments.

Examples

A good way to understand reinforcement learning is to consider some of the examples and possible applications that have guided its development.

- A master chess player makes a move The choice is informed both by planning-anticipating possible replies and counterreplies - and by immediate, intuitive judgments of the desirability of particular positions and moves.
- An adaptive controller adjusts parameters of a petroleum refinery's operation in real time. The controller optimizes the yield/cost/quality trade-off on the basis of specified marginal costs without sticking strictly to the set points originally suggested by engineers.
- A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 20 miles per hour.
- A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find and its way back to its battery recharging station. It makes its decision based on the current charge level of its battery and how quickly and easily it has been able to find the recharger in the past.

These examples share features that are so basic that they are easy to over-look. All involve $interaction$ between an active decsiion-making agent and its environment, within which the agent seeks to achieve a $goal$ despite $uncertainty$ about its environment.

The agent's actions are permitted to affect the future state of the environment and thereby affecting the options and opportunities available to the agent at later times. At the same time, in all these examples the effects of actions cannot be fully predicted., thus the agent must monitor its environment frequently and react appropriately.

Elements of Reinforcement Learning

Beyond the agent and the environment, one can identify four main subelements of a reinforcement learning system.

1. A $policy$
2. A $reward\ signal$
3. A $value\ function$
4. A $model$ of the environment

$Policy$

A $policy$ defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic.

$Reward$

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. In a biological system, we might think of rewards as analogous to the experiences of pleasure or pain. The reward sent to the agent at any time depends on the agent's current action and the current state of the agent's environment. The agent cannot alter the process that does this. The only way the agent can influence the reward signal is through its actions, which can have a direct effect on reward, or an indirect effect through change the environment's state. **The reward signal is the primary basis for altering the policy.**

If an action selection by the policy is followed by low reward, then the policy may be changes to select some other action in that situation in the future.

in general, rewards signals may be stochastic functions of the state of the environment and the actions taken.

$Value\ Function$

A $value\ function$ specifies what is good in the long run. 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 $long\text{-}term$ desirability of states after taking into account the states that are likely to follow, and the rewards available in those states.

For example, a state might always yield a low immediate rewards but still have a high value because it is regularly followed by other states that yield high rewards.

Rewards are in a sense **primary** , whereas values, as predictions of rewards, are **Secondary**.

Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations 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.

The fourth and final element of some reinforcement learning systems is a $model$ of the environment. This is something that mimics the behaviour of the environment , or more generally , that allows inferences to be made about how the environment will behave.

An Extended Example : Tic-Tac-Toe

To illustrate the general idea of reinforcement learning and contrast it with other appraoches , we next consider a single example

Consider the familiar child's game of tic-tac-toe. Two players take turns playing on a three-by-three board. One player plays $X's$ and the other Os until one player wins by placing three marks in a row, horizontally, vertically or diagonally, as the X player has in this game.

X	O	O
O	X	X
		X

For the moment, in fact, let us consider draws and losses to be equally bad for us. How might we construct a player that will find the imperfections in this opponent's play and learn to maximize its chances of winning ?

An evolutionary method applied to this problem would directly search the space of possible policies for one with high probability of winning against the opponent.

Here, a policy is a rule that tells the player what move to make for every state of the game - every possible configuration of Xs and Os on three-by-three board. For each policy considered, an estimate of its winning probability would be obtained by playing some number of games against the opponent. This evaluation would then direct which policy or policies were considered next. A typical evolutionary method would hill-climb in policy space , successively generating and evaluating policies in an attempt to obtain incremental improvements.

Here is how the tic-tac-toe problem would be approached with a method making use of a value function.

- First we set up a table of numbers , one for each possible state of the game. Each number will be the latest estimate of the probability of our winning from that state. We treat this estimate as the state's $value$, and the whole table is the learned value function. State A has higher value than state B, or is considered "better" than state B, if the current estimate of the probability of our winning from A is higher than its is from B.

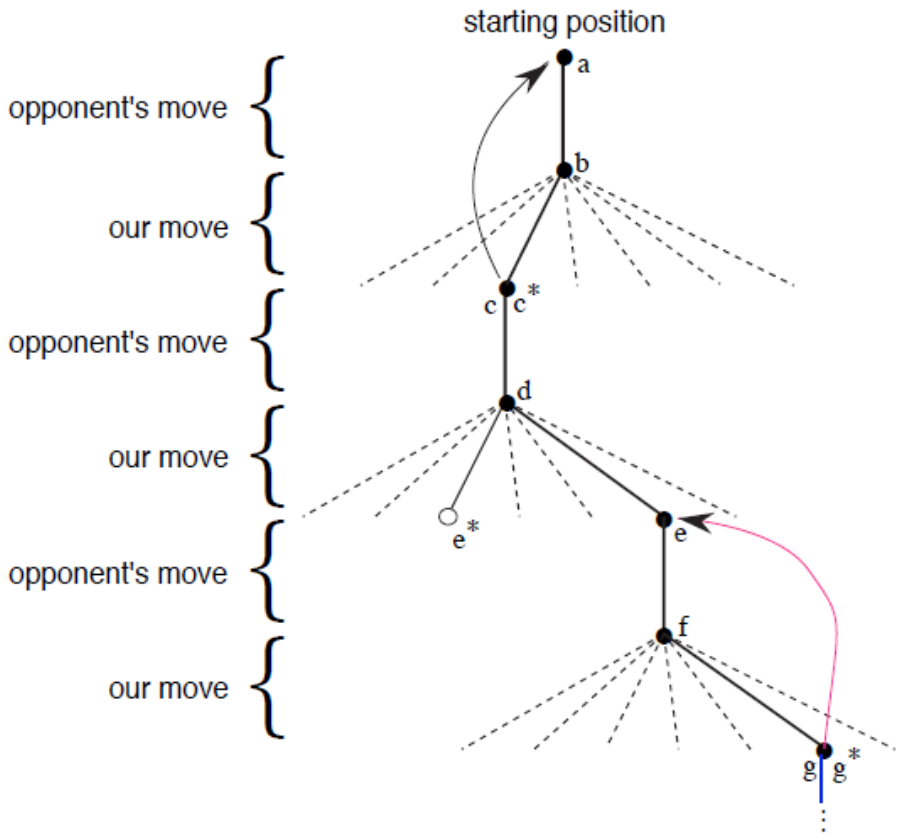

Figure 1.1: A sequence of tic-tac-toe moves. The solid lines represent the moves taken during a game; the dashed lines represent moves that we (our reinforcement learning player) considered but did not make. Our second move was an exploratory move, meaning that it was taken even though another sibling move, the one leading to e*, was ranked higher. Exploratory moves do not result in any learning, but each of our other moves does, causing *backups* as suggested by the curved arrows and detailed in the text.

- Assuming we always play Xs , then for all states with three Xs in a row the probability of winning is 1, because we have already onw. Similarly, for all states with three Os in a row, or that are "filled up" , the correct probability is 0, as we cannot win from them. We set the initial values of all the other states to 0.5 , representing a guess that we have a 50% chance of winning.
- We play games against the opponent. To select our moves we examine the states that would result from each of our possible moves (one for each blank space on the board)

and look up their current values in the table. Most of the time we move $greedily$, selecting the move that leads to the state with greatest value, that is , with the highest estimated probability of winning. Occasionally , however, we select randomly from among the other moves instead. These are called $exploratory$ moves because they cause us to experience states that we might otherwise never see.

- 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 closed 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.

  If we let s denote the state before the greedy move, and s' the state after the move, then the update to the estimated value of s, denoted $V(s)$, can be written as

  $$V(s) \longleftarrow V(S) + \alpha[V(s') - V(s)],$$

  where α is a small positive fraction call 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 , $V(s') - V(s)$, betweent estimates at two different times.

This method described above performs quite well on its task. For exmaple, if the step-size parameter is reduced properly over time, this method converges, for any fixed opponent , to the true probabilities of winning from each state given optimal play by our player.Furthermore, the moves then taken(except on exploratory moves) are in fact the optimal moves against the opponent. In other words, the method converges to an optimal policy for playing the game.

Summary

Reinforcement learning is a computational approach to understanding an automating goal-directed learning and decision making. It is distinguished from other computational approaches by its emphasis on learning by an agent from direct interaction with its environment, without relying on exemplary supervision or complete models of the environment.

Reinforcement learning uses a formal framework defining the interaction between a learning agent and its environment in terms of states ,actions and rewards. This framework is intended to be a simple way of representing features of the aritifical intelligence problem. These features include a sense of cause and effect, as sense of uncertaining and nondeterminism , and the existence of explicity goals.