Reinforcement Learning

TD-Learning

Value Functions

Before Temporal Difference Learning can be explained, it is necessary to start with a basic understanding of Value Functions. Value Functions are state-action pair functions that estimate how good a particular action will be in a given state, or what the return for that action is expected to be. The following notation is used:

$V^{\pi}(s)$ - the value of a state s under policy π. The expected return when starting in s and following π thereafter

$Q^{\pi}(s, a)$ - the value of taking action a in state s under a policy π. The expected return when starting from s taking the action a and thereafter following policy π. From here on, this will be referred to as a Q-value.

The problem at hand is estimating these value functions for a particular policy. The reason we want to estimate these value functions is so that they can be used to accurately choose an action that will provide the best total reward possible, after being in that given state.

Temporal Difference Learning

Temporal Difference (TD) Learning methods can be used to estimate these value functions. If the value functions were to be calculated without estimation, the agent would need to wait until the final reward was received before any state-action pair values can be updated. Once the final reward was received, the path taken to reach the final state would need to be traced back and each value updated accordingly. This can be expressed formally as:

$$V(s_t) \leftarrow V(s_t) + \alpha [R_t - V(s_t)]$$

where s_t is the state visited at time t, R_t is the reward after time t and α is a constant parameter

On the other hand, with TD methods, an estimate of the final reward is calculated at each state and the state-action value updated for every step of the way. Expressed formally:

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

where $\mathbf{r_{t+1}}$ is the observed reward at time **t+1**

The TD method is called a "bootstrapping" method, becuase the value is updated partly using an existing estimate and not a final reward.

On-Policy and Off-Policy Learning

On-Policy Temporal Difference methods learn the value of the policy that is used to make decisions. The value functions are updated using results from executing actions determined by some policy. These policies are usually "soft" and non-deterministic. The meaning of "soft" in this sense, is that it ensures there is always an element of exploration to the policy. The policy is not so strict that it always chooses the action that gives the most reward. Three common policies are used, ε-soft, ε-greedy and softmax. These are explained in the section below.

Off-Policy methods can learn different policies for behaviour and estimation. Again, the behaviour policy is usually "soft" so there is sufficient exploration going on. Off-policy algorithms can update the estimated value functions using hypothetical actions, those which have not actually been tried. This is in contrast to on-policy methods which update value functions based strictly on experience. What this means is off-policy algorithms can separate exploration from control, and on-policy algorithms cannot. In other words, an agent trained using an off-policy method may end up learning tactics that it did not necessarily exhibit during the learning phase.

Action Selection Policies

As mentioned above, there are three common policies used for action selection. The aim of these policies is to balance the trade-off between exploitation and exploration, by not always exploiting what has been learnt so far.

> ε-greedy - most of the time the action with the highest estimated reward is chosen, called the greediest action. Every once in a while, say with a small probability ε, an action is selected at random. The action is selected uniformly, independant of the action-value estimates. This method ensures that if enough trials are done, each action will be tried an infinite number of times, thus ensuring optimal actions are discovered.
>
> ε-soft - very similar to ε-greedy. The best action is selected with probability 1 - ε and the rest of the time a random action is chosen uniformly.
>
> softmax - one drawback of ε-greedy and ε-soft is that they select random actions uniformly. The worst possible action is just as likely to be selected as the second best. Softmax remedies this by assigning a rank or weight to each of the actions, according to their action-value estimate. A random action is selected with regards to the weight associated with each action, meaning the worst actions are unlikely to be chosen. This is a good approach to take where the worst actions are very unfavourable.

It is not clear which of these policies produces the best results overall. The nature of the task will have some bearing on how well each policy influences learning. If the problem we are trying to solve is of a game playing nature, against a human opponent, human factors may also be influencial.

Next...

Q-Learning and Sarsa explained, plus an example illustrating the difference between On-Policy and Off-Policy algorithms.

Previous page Next page