Reinforcement Learning

Algorithms

Q-Learning

Q-Learning is an Off-Policy algorithm for Temporal Difference learning. It can be proven that given sufficient training under any ε-soft policy, the algorithm converges with probability 1 to a close approximation of the action-value function for an arbitrary target policy. Q-Learning learns the optimal policy even when actions are selected according to a more exploratory or even random policy. The procedural form of the algorithm is:

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., ε-greedy)
      Take action a, observe r, s'
      $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
      s <-- s';
   until s is terminal
```

The parameters used in the Q-value update process are:

α - the learning rate, set between 0 and 1. Setting it to 0 means that the Q-values are never updated, hence nothing is learned. Setting a high value such as 0.9 means that learning can occur quickly.

γ - discount factor, also set between 0 and 1. This models the fact that future rewards are worth less than immediate rewards. Mathematically, the discount factor needs to be set less than 0 for the algorithm to converge.

$\max_{a}$ - the maximum reward that is attainable in the state following the current one. i.e the reward for taking the optimal action thereafter.

This procedural approach can be translated into plain english steps as follows:

1. Initialize the Q-values table, Q(s, a).
2. Observe the current state, s.
3. Choose an action, a, for that state based on one of the action selection policies explained here on the previous page (ε-soft, ε-greedy or softmax).
4. Take the action, and observe the reward, r, as well as the new state, s'.
5. Update the Q-value for the state using the observed reward and the maximum reward possible for the next state. The updating is done according to the forumla and parameters described above.
6. Set the state to the new state, and repeat the process until a terminal state is reached.

Sarsa

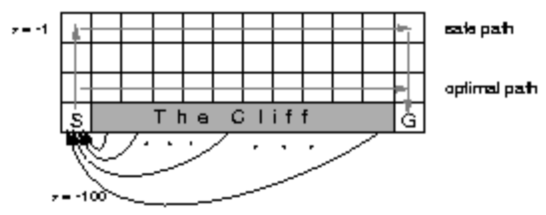
The Sarsa algorithm is an On-Policy algorithm for TD-Learning. The major difference between it and Q-Learning, is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name Sarsa actually comes from the fact that the updates are done using the quintuple Q(s, a, r, s', a'). Where: s, a are the original state and action, r is the reward observed in the following state and s', a' are the new state-action pair. The procedural form of Sarsa algorithm is comparable to that of Q-Learning:

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Choose a from s using policy derived from Q
         (e.g., $\varepsilon$-greedy)
   Repeat (for each step of episode):
      Take action a, observe r, s'
      Choose a' from s' using policy derived from Q
         (e.g., $\varepsilon$-greedy)
      $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
      s <-- s'; a <-- a';
   until s is terminal
```

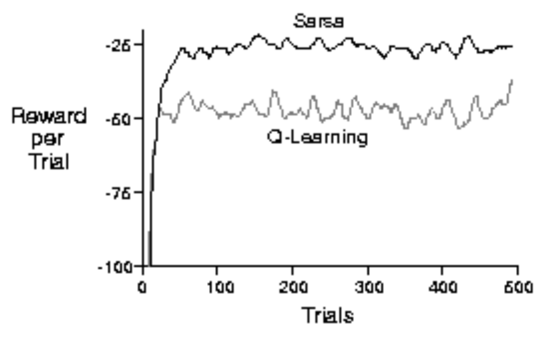
As you can see, there are two action selection steps needed, for determining the next state-action pair along with the first. The parameters α and γ have the same meaning as they do in Q-Learning.

Example

To highlight the difference between Q-Learning and Sarsa, an example from [1] will be used. They took the cliff world shown below:



The world consists of a small grid. The goal-state of the world is the square marked G on the lower right-hand corner, and the start is the S square in the lower left-hand corner. There is a reward of negative 100 associated with moving off the cliff and negative 1 when in the top row of the world. Q-Learning correctly learns the optimal path along the edge of the cliff, but falls off every now and then due to the ε-greedy action selection. Sarsa learns the safe path, along the top row of the grid because it takes the action selection method into account when learning. Because Sarsa learns the safe path, it actually receives a higher average reward per trial than Q-Learning even though it does not walk the optimal path. Here is a graph showing the reward per trial for both Sarsa and Q-Learning:



Next...

Test drive the algorithms and Action Selection policies using our applet.

Previous page Next page