Q-Learning

MICHAEL WOLLOWSKI

Introduction

In Q-Learning, an agent tries to learn a policy from what it learned by interacting with its environment.

So far, an agent learned policies before it even took a step.

Now, it will explore its world and as it does so, it will update its policy.

This is a form of temporal difference learning.

An agent learns an action-utility function, or Q-function.

Q-Learning

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
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'
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$;
  until S is terminal

Figure 6.12: Q-learning: An off-policy TD control algorithm.

Algorithm source: Russell and Norvig: AIMA 2nd ed.

Exploration vs. Exploitation

Let $f(u, n)$ be an *exploration function*.

It determines how greed (preference for high values of utility u) is traded off against curiosity (preference for actions that have not been tried often and have a low frequency count n.)

The function should be increasing in u and decreasing in n.

A simple definition is:
$f(u, n) = R^+$, if $n < N_e$
u otherwise

- R^+ is the expected reward.
- N_e is a fixed parameter.

Q-Learning with Exploitation

function Q-LEARNING-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  persistent: Q, a table of action values indexed by state and action, initially zero
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if TERMINAL?(s') **then** $Q[s', None] \leftarrow r'$
  if s is not null **then**
    increment $N_{sa}[s, a]$
    $Q[s,a] \leftarrow Q[s,a] + \alpha(N_{sa}[s,a])(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
  $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[s',a'], N_{sa}[s',a']), r'$
  return a

Algorithm source: Russell and Norvig: AIMA 2nd ed.