Dynamic Programming

Wednesday, June 17, 2020 10:49 AM

Policy Evaluation:

How to compute the state-value function v_π for an arbitrary policy π

Iterative Policy Evaluation:

$$v_{k+1}(s) \equiv E_\pi\big[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s\big] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma v_k(s')\big]$$

Where $v_0, v_1, v_2 \cdots$ is a sequence of value functions that will converge to the true state-value function for a given policy and v_0 is chosen arbitrarily but with the condition that the terminal state (if it exists) must be given value 0

Note that this works by setting the value of each state in the next iteration to the expected value of the state in the previous iteration (knowing the values of the successor states in the previous iteration)

This process is called an expected update

This is because the update is based on an expectation over all possible next states as opposed to a sample next state

Note that $v_k = v_\pi$ is a stationary point

That is to say that the true state-value function will return itself for the next step and the function will stop iterating

This can be used to stop this algorithm in practical computation (for example check the maximum amount a state changes by between iterations ($\max_s \big|v_{k+1}(s) - v_k(s)\big|$) and terminate when this gets below a certain value

This is guaranteed to converge as $k \to \infty$ given the same conditions that guarantee v_π exists

That is to say that if a true state-value function exists, this algorithm will find it

Note that in practical computation this algorithm can be implemented using a single array updated in place

Policy Improvement Theorem:

$$q_\pi\big(s, \pi'(s)\big) \geq v_\pi(s)$$

If this is true for all states then the policy π' is as good as or better then policy π

Alternative statement if the above is true:

$$v_{\pi'}(s) \geq v_\pi(s)$$

Note that if the strict inequality holds for the first condition it will also hold for this condition

Greedy Policy:

$$\pi' \equiv \operatorname*{argmax}_a q_\pi(s, a) = \operatorname*{argmax}_a E\big[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a\big]$$

$$= \operatorname*{argmax}_a \sum_{s',r} p\big(s', r|s, a\big)\big[r + \gamma v_\pi(s')\big]$$

Where v_π is the value function of a previous policy

Note that if this policy is as good as but not better than the old policy then this is the Bellman optimality equation and both policies are optimal

Policy Improvement:

The process of improving on an original policy by making it greedy with respect to the value function of the original policy

This is setting the new policy equal to the greedy policy above

Policy Iteration:

$$\pi_0 \overset{E}{\to} v_{\pi_0} \overset{I}{\to} \pi_1 \overset{E}{\to} v_{\pi_1} \overset{I}{\to} \pi_2 \overset{E}{\to} \cdots \overset{I}{\to} \pi_* \overset{E}{\to} v_*$$

This is the process of alternative between policy evaluation $\left(\overset{E}{\to}\right)$ and improvement $\left(\overset{I}{\to}\right)$ until an optimal policy is reached

Note that the initial values for the value function in each policy evaluation procedure are the values of the value function of the previous policy

Value Iteration:

$$v_{k+1}(s) \equiv \max_{a} E\left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a\right] = \max_{a} \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_k(s')\right]$$

This is the case where policy evaluation is stopped after one update of each state

This is guaranteed to converge to v_* under the same conditions that guarantee the existence of v_*

One use of this is for asynchronous dynamic programming, where the value of a state is updated when the agent encounters that state

Generalized Policy Iteration:

The general idea of letting policy evaluation and policy iteration interact, independent of the granularity and other details of the two processes

As long as both processes continue to update all states as the number of time steps goes to infinity, convergence to the optimal value function and policy will usually occur

In general, the value function stabilizes only when it is consistent with the current policy and the policy stabilizes only when it is greedy with respect to the current value function. If both conditions are met then the current policy is greedy with respect to its own evaluation policy and the Bellman optimality equation holds

In the worst case, the time that DP (dynamic programming) methods take to find an optimal policy is polynomial in the number of states and actions

Asynchronous DP methods are usually applied to problems with large state spaces

Bootstrapping:

The idea of updating estimates based on other estimates