Markov Decision Processes

Anshul Choudhary

8 June, 2025

Solution to Exercises

Question

Derive optimal bellman equations for value and action value functions.

Answer

We know the value function is given by:-

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s'))$$

Now a policy can be considered optimal if it greedily picks the action at each state which gives the maximum Expected return

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)(r + \gamma v_*(s'))$$

here v_* is the optimal value function, The policies which satisy this system of equations can be called optimal policies. ,though defining it this way , it is not clear whether a policy with such a value function even exists but we will prove it later.
Similarly the action value function:-

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s',a') \right]$$

again choosing greedily we get the optimal action value function as:-

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a) \left[r + \gamma \max_{a'} q_*(s',a') \right]$$

Question

What is optimality?

Answer

Optimality in context of MDPs refers to picking the action(or policy) with the highest expected return in a specific state(or environment).

Question

What is expectation and how does it relate to bellman equations?

Answer

Expected value of a random variable is the mean of all possible values of the variable weighted by the probabilities of those values. this is the bellman equation for value function in terms of Expectation-

$$v_\pi = E_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

here R_{t+1} is the reward received at time $t + 1$ and G_{t+1} is the cumulative reward received from time t+1 till termination or infinity.
So basically the value function tells us the expected reward we will get following a policy π. The reason we use expectation is because mostly we are dealing with stochastic environments which react differently even when following same actions, so the natural thing to look at is Expectation which captures all possibilities.

Exercise from Chapter 3 Sutton and Barto:-

Question

Give an equation for v_* in terms of q_*.

Answer

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)(r + \gamma v_*(s'))$$

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right]$$

The action-value function $q_*(s, a)$ gives the maximum expected return achievable by taking action a in state s, and then following an optimal policy thereafter.
The value function $v_*(s)$ gives the maximum expected return starting from state s, following the optimal policy at all states. Therefore, it is simply the maximum over all possible actions of $q_*(s, a)$:
thus-

$$v_*(s) = \max_a q_*(s, a)$$

Question

Give an equation for q_* in terms of v_* and the four-argument p.

Answer

As said earlier $q_*(s,a)$ means taking the action a and then following the optimal policy from the resulting state, this idea can be easily written in terms of the four argument p function which will tell us about the possible future states and reward received after taking action a.

$$q_*(s,a) = \sum_{s',r} P(s',r|s,a)(r + \gamma v_*(s'))$$

Question

Give an equation for π_* in terms of q_* .

Answer

We have to derive the optimal policy given the optimal action value function.So we can simply iterate over all actions in any given state and check at which action the action-value function takes its maximum and then that action will be the most optimal one, here it is possible that more than one action give the same value of the action-value function so we can either assign a probability of 1 to any one of them and that would give us a deterministic policy or we could assign equal probabilities to all such actions. the latter can be easily represented in terms of one-hot vectors

$$\pi_*(a \mid s) = \frac{\mathbf{1}\left\{a \in \arg\max_{a'} q_*(s,a')\right\}}{\sum_{a''} \mathbf{1}\left\{a'' \in \arg\max_{a'} q_*(s,a')\right\}}$$

Question

Give an equation for π_* in terms of v_* and the four-argument p.

Answer

We just have to substitute the expression of q_* that we got in terms of v_* in 3.26 in the formula we got in 3.27.

Question

Rewrite the four Bellman equations for the four value functions (v_π , v_* , q_π , and q_*) in terms of the three argument function p (3.4) and the two-argument function r (3.5).

Answer

this is in terms of the 4 argument p

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s'))$$

the inner sum can be rewritten as -

$$\sum_{s',r} p(s', r|s, a)r + \gamma \sum_{s',r} p(s', r|s, a)v_\pi(s')$$

the first summation is just $r(s, a)$ and since the second sums over all r it is equal to $\sum_{s'} p(s'|s, a)v_\pi(s')$
so finally the expression becomes-

$$v_\pi(s) = \sum_a \pi(a|s) \left(r(s, a) + \gamma \sum_{s'} p(s'|s, a)v_\pi(s') \right)$$

Similarly,

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \sum_{a'} \pi(a'|s')q_\pi(s', a')$$

For $v_*(s)$ the inner sum from $v_\pi(s)$ will be the same but instead of weighting by some policy we will simply take the max of that sum over all actions.

$$v_*(s) = \max_a \left(r(s, a) + \gamma \sum_{s'} p(s'|s, a)v_*(s') \right)$$

Similarly,

$$q_*(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} q_*(s', a')$$

Question

What actually is GPI?

Answer

Generalized Policy Iteration (GPI) is a general framework describing the interaction between two processes: policy evaluation and policy improvement.
In policy iteration, we fully evaluate the value function for a given policy and then improve the policy greedily based on this value function. In value iteration, we repeatedly apply the Bellman optimality operator to approximate the optimal value function, implicitly improving the policy.
GPI generalizes these ideas: it allows the policy evaluation and policy improvement steps to proceed in parallel or interleaved, without requiring full convergence of either step before the other proceeds. if both policy evaluation and policy improvement are done sufficient number of times the algorithm converges.

Question

What is the complete and rigorous proof of their convergence?

Answer

First lets see why policy iteration and value iteration converge.This will help us understand the general convergence of Generalized Policy Iteration (GPI).

Policy Iteration:

In each step, we compute the value function for the current policy, and then improve the policy by making it greedy with respect to this value function. The *Policy Improvement Theorem* ensures that this greedy step produces a policy that is at least as good as the previous one, and strictly better unless the current policy is already optimal.

Since there are only finitely many policies (if the action space is finite), and the value function strictly improves unless the policy is already optimal, this process must eventually converge to the optimal policy.

Value Iteration:

We start with any arbitrary value function and iteratively apply the Bellman optimality operator:

$$v_{k+1} = T^*(v_k)$$

where T^* is the Bellman optimality operator.
The Bellman operator is a map from value functions to value functions, if we can find a fixed point for this map such that $T^*(v) = v$ then we will be done because this value function satisfies the Bellman optimality equations(after all the Bellman optimality operator is just the Bellman optimality equations represented as a map). The Bellman optimality operator is a contraction mapping under the max norm (sup-norm), with contraction factor $\gamma < 1$.

By the Banach Fixed Point Theorem , a contraction mapping on a complete metric space has a unique fixed point. Therefore, T^* has a unique fixed point v_*, which is the optimal value function. Moreover, repeated application of T^*, i.e., value iteration, is guaranteed to converge to v_*.

Generalized Policy Iteration (GPI):

GPI is a general framework where the policy evaluation and policy improvement processes interact — they do not need to run to full convergence at each step.

GPI is basically interleaving value iteration and policy iteration and since each of these methods separately approach towards the optimum at every step(they improve at every step), thus even after interleaving these we will always move towards the optimum, this is the essential property that guarantees convergence; the fact that the policy is monotonically improved.

Question

What is the time and memory complexity for each of the algorithms? Where are each of them particularly useful?

Answer

The time complexity for each iteration of policy and value iteration is $O(|\mathcal{S}|^2|\mathcal{A}|)$ because in every iteration we have to update the value function for all states so that gives a factor of $|\mathcal{S}|$ and to update each state we have to look at all other states in the observation space and actions leading to those states which gives the factor of $|\mathcal{S}||\mathcal{A}|$. This is strictly an upper bound because we are assuming at each state all other states are reachable and also all actions from the action space are also available, this might not always be true for example in a maze like environment we can possibly only go to 4 neighboring squares or maybe 8 depending if diagonal movement is allowed or not, in any case if the grid is big then this will be quite small comparatively.

Number of iteration required in value iteration for convergence:-

As for number of iterations required to converge, to converge exactly it would take infinite iterations but this doesn't mean we cannot reach the optimal policy , we don't need the exact value function, since we are going to use argmax later anyways so even if the value function is within some tolerable range of the true function it would suffice to get the optimal policy, so let us assume that we want the error to be no more than ϵ. since Bellman operator is a contraction we can say-

$$\|v_k - v_*\| \le \gamma^k \|v_0 - v_*\|$$

To achieve an error $\le \epsilon$, we require:

$$\gamma^k \|v_0 - v_*\| \le \epsilon$$

Taking logarithms:

$$k \ge \frac{\log(\epsilon/\|v_0 - v_*\|)}{\log(\gamma)} = \frac{\log(1/\epsilon) + \log(\|v_0 - v_*\|)}{\log(1/\gamma)}$$

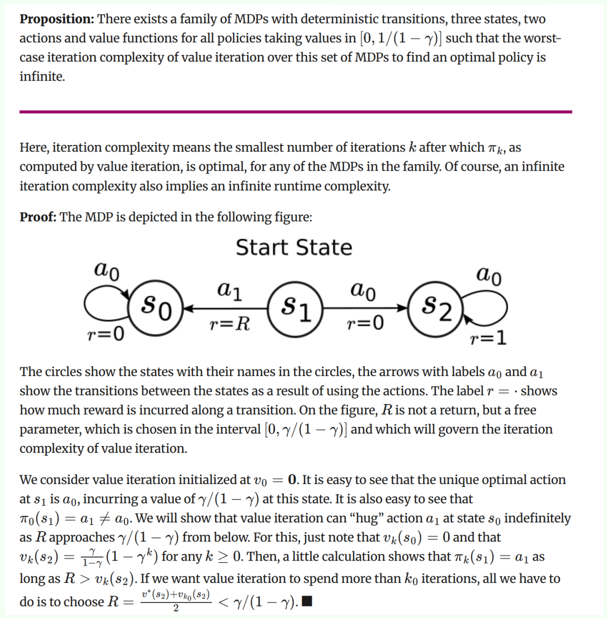
Thus, the number of iterations is:

$$k = O\left(\frac{\log(1/\epsilon)}{\log(1/\gamma)}\right)$$

It can be seen from the above equation and is also evident logically that the number of iterations will increase as ϵ decreases. It is also interesting to note that as gamma decreases the number of iterations reduce, the reason for this is that the effect of states far away become less and less important and only very nearby states matter, this decreases the complexity of assignment and error decreases faster but if γ was higher the small errors in estimating distant rewards would compound over time and take more iterations to converge.

Number of iteration required in policy iteration for convergence:-
This article on policy iteration is a very interesting read , it talks very carefully about time complexity and number of elementary arithmetic operation required to converge.To summarize some of the interesting points:-

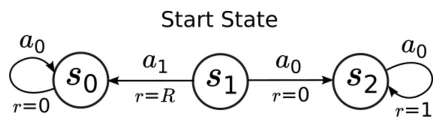
1. firstly there are two ways to implement the policy evaluation step , one is the iterative way and one is by observing the fact that the system is simply a system of linear equations which will have a complexity of $O(|S^{2.37}|)$. A single iteration of the iterative method is $O(|S^2||A|)$ but the number of iteration required to converge can grow with the value of ϵ ie how much error we can tolerate in the solution. So in cases where it is taking too many iteration to converge we can fall back to the matrix method.
2. Secondly the number of iteration for policy iteration does not grow very rapidly with ϵ like it does with value iteration and this is a serious pitfall for value iteration; the article goes on to prove the existence of MDPs for which we can control the number of iteration they will need to converge.



Proposition: There exists a family of MDPs with deterministic transitions, three states, two actions and value functions for all policies taking values in $[0, 1/(1-\gamma)]$ such that the worst-case iteration complexity of value iteration over this set of MDPs to find an optimal policy is infinite.

Here, iteration complexity means the smallest number of iterations k after which π_k, as computed by value iteration, is optimal, for any of the MDPs in the family. Of course, an infinite iteration complexity also implies an infinite runtime complexity.

Proof: The MDP is depicted in the following figure:



The circles show the states with their names in the circles, the arrows with labels a_0 and a_1 show the transitions between the states as a result of using the actions. The label $r = \cdot$ shows how much reward is incurred along a transition. On the figure, R is not a return, but a free parameter, which is chosen in the interval $[0, \gamma/(1-\gamma)]$ and which will govern the iteration complexity of value iteration.

We consider value iteration initialized at $v_0 = \mathbf{0}$. It is easy to see that the unique optimal action at s_1 is a_0, incurring a value of $\gamma/(1-\gamma)$ at this state. It is also easy to see that $\pi_0(s_1) = a_1 \neq a_0$. We will show that value iteration can “hug” action a_1 at state s_0 indefinitely as R approaches $\gamma/(1-\gamma)$ from below. For this, just note that $v_k(s_0) = 0$ and that $v_k(s_2) = \frac{\gamma}{1-\gamma}(1-\gamma^k)$ for any $k \geq 0$. Then, a little calculation shows that $\pi_k(s_1) = a_1$ as long as $R > v_k(s_2)$. If we want value iteration to spend more than k_0 iterations, all we have to do is to choose $R = \frac{v^*(s_2)+v_{k_0}(s_2)}{2} < \gamma/(1-\gamma)$. ■

This is still an extreme case but even in general cases , policy iteration will converge in fewer iterations compared to value iteration(though the cost of a single iteration is more in policy iteration than in value iteration). But still in most cases policy iteration would be the recommended method. Space complexity should be $O(|\mathcal{S}|)$ since we have to store value function for all states.

Books and Resources I used

1.Bellman operator

2.Time complexity of algorithms

3.Basic introduction to metric spaces and contraction mappings

4.Proof that Bellman Operator is a contraction mapping

5.Banach Fixed Point Theorem

6..Effect of γ on learning