Dynamic Programming:

- **Dynamic programming (DP) is a technique for solving complex problems.** In DP, instead of solving a complex problem as a whole, we break the problem into simple sub-problems, then for each subproblem,we compute and store the solution.

- If the same subproblem occurs, we don't recompute; instead, we use the already computed solution. Thus, DP helps in drastically minimizing the computation time.

Now, we will learn about two important methods that use DP to find the optimal policy. The two methods are:

1. Value iteration
2. Policy iteration

- Note that dynamic programming is a **model-based method** meaning that it will help us to find the optimal policy only when the model dynamics (transition probability) of the environment are known.

- If we don't have the model dynamics, we cannot apply DP methods.

Value Iteration

In the value iteration method, we try to find the optimal policy. We learned that the optimal policy is the one that tells the agent to perform the correct action in each state. In order to find the optimal policy, first, we compute the **optimal value function** and once we have the optimal value function, we can use it to derive the optimal policy.

We know that the Bellman equation and the relationship between the value and the Q function.

$$V^*(s) = \max_a Q^*(s, a)$$

Thus, we can compute the optimal value function by just taking the maximum over the optimal Q function. So, in order to compute the value of a state, we compute the Q value for all state-action pairs. Then, we select the maximum Q value as the value of the state.

Let's understand this with an example.

State	Action	Value
s_0	0	2.7
s_0	1	3
s_1	0	4
s_1	1	2

Q values of all possible state-action pairs

→ Say we have two states, s0 and s1, and we have two possible actions in these states; let the actions be 0 and 1. And corresponding Q value for all possible state-action pairs.

State	Value
s_0	3
s_1	4

Optimal state values

→ Then, in each state, we select the maximum Q value as the optimal value of a state.we can use it to extract the optimal policy.

The value iteration algorithm with example

Let's say we are in state A and our goal is to reach state C without visiting the shaded state B, and say we have two actions, 0 — left/right, and 1 — up/down.

→ The optimal policy here is the one that tells us to perform action 1 in state A so that we can reach C without visiting B.

State (s)	Action (a)	Next State (s')	Transition Probability $P(s'\|s,a)$ or $P^a_{ss'}$	Reward Function $R(s,a,s')$ or $R^a_{ss'}$
A	0	A	0.1	0
A	0	B	0.8	-1
A	0	C	0.1	1
A	1	A	0.1	0
A	1	B	0.0	-1
A	1	C	0.9	1

model dynamics of state **A**

Step 1 – Compute the optimal value function

we compute the Q value for all state-action pairs and then we select the maximum Q value as the value of a state

The Q value for a state s and action a can be computed as:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$$

For notation simplicity, we can denote $P(s'|s,a)$ by $P^a_{ss'}$ and $R(s,a,s')$ by $R^a_{ss'}$ and rewrite the preceding equation as:

$$Q(s,a) = \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V(s')\right]$$

- Using the preceding equation to compute the Q function, we need the transition probability , the reward function , and the value of the next state . The model dynamics provide us with the transition probability and the reward function .

But what about the value of the next state??

- We don't know the value of any states yet. So, we will initialize the value function (state values) with random values or zeros to compute the Q function.

State	Value
A	0
B	0
C	0

Initial value table

discount factor $\gamma = 1$

Iteration 1:

Let's compute the Q value for state A and action 0:

$$\begin{aligned} Q(A,0) &= P^0_{AA}[R^0_{AA} + \gamma V(A)] + P^0_{AB}[R^0_{AB} + \gamma V(B)] + P^0_{AC}[R^0_{AC} + \gamma V(C)] \\ &= 0.1(0+0) + 0.8(-1+0) + 0.1(1+0) \\ &= -0.7 \end{aligned}$$

Let's compute the Q value for state A and action 1:

$$\begin{aligned} Q(A,1) &= P^1_{AA}[R^1_{AA} + \gamma V(A)] + P^1_{AB}[R^1_{AB} + \gamma V(B)] + P^1_{AC}[R^1_{AC} + \gamma V(C)] \\ &= 0.1(0+0) + 0.0(-1+0) + 0.9(1+0) \\ &= 0.9 \end{aligned}$$

After computing the Q values for both the actions in state A, we can update the Q table

State	Action	Value
A	0	- 0.7
A	1	0.9
B	0	
B	1	
C	0	
C	1	

Q table

$$V^*(s) = \max_a Q^*(s, a)$$

We learned that the optimal value of a state is just the max of the Q function. By looking at table, we can say that the value of state A, V(A), is Q(A, 1) since Q(A, 1) has a higher value than Q(A, 0). Thus, V(A) = 0.9.

We can update the value of state A in our value table

State	Value
A	0.9
B	0
C	0

Updated value table

- Similarly, in order to compute the value of state B, V(B), we compute the Q value of Q(B, 0) and Q(B, 1) and select the highest Q value as the value of state B.
- In the same way, to compute the values of other states, we compute the Q value for all state-action pairs and select the maximum Q value as the value of a state.
- After computing the value of all the states, our updated value table is the result of the first iteration:

State	Value
A	0.9
B	- 0.2
C	0.5

Value table from iteration 1

However, the value function (value table) obtained as a result of the first iteration is not an optimal one as we computed the Q function based on the randomly initialized state values.In short,

$$V^*(s) = \max_a Q^*(s,a)$$

$$\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$$

Random Values

But in the second iteration, we use the updated state values (value table) obtained from the first iteration as the following shows:

$$V^*(s) = \max_a Q^*(s,a)$$

$$\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$$

Use state values from 1st iteration

Iteration 2

Let's compute the Q value of state A. Remember that while computing the Q value, we use the updated state values from the previous iteration.

First, let's compute the Q value of state **A** and action 0:

$$
\begin{aligned}
Q(A,0) &= P^0_{AA}[R^0_{AA} + \gamma V(A)] + P^0_{AB}[R^0_{AB} + \gamma V(B)] + P^0_{AC}[R^0_{AC} + \gamma V(C)] \\
&= 0.1(0 + 0.9) + 0.8(-1 - 0.2) + 0.1(1 + 0.5) \\
&= -0.72
\end{aligned}
$$

Now, let's compute the Q value for state **A** and action 1:

$$
\begin{aligned}
Q(A,1) &= P^1_{AA}[R^1_{AA} + \gamma V(A)] + P^1_{AB}[R^1_{AB} + \gamma V(B)] + P^1_{AC}[R^1_{AC} + \gamma V(C)] \\
&= 0.1(0 + 0.9) + 0.0(-1 - 0.2) + 0.9(1 + 0.5) \\
&= 1.44
\end{aligned}
$$

As we may observe, since the Q value of action 1 in state A is higher than action 0, the value of state A becomes 1.44. Similarly, we compute the value for all the states and update the value table.

State	Value
A	1.44
B	- 0.50
C	1.0

Value table from iteration 2

Iteration 3: So, we use the updated state values from iteration 2 to compute the Q value.

State	Value
A	1.94
B	- 0.70
C	1.3

Value table from iteration 3

So, we repeat these steps for many iterations until we find the optimal value function.But **how can we understand whether we have found the optimal value function or not??**

- When the value function (value table) does not change over iterations or when it changes by a very small fraction, then we can say that we have attained convergence, that is, we have found an optimal value function.

how can we find out whether the value table is changing or not changing from the previous iteration?

- We can calculate the difference between the value table obtained from the previous iteration and the value table obtained from the current iteration. If the difference is very small—say, the difference is less than a very small threshold number—then we can say that we have attained convergence as there is not much change in the value function.

State	Value
A	1.94
B	- 0.70
C	1.3

Value table from iteration 4

→ let's suppose value table obtained as a result of iteration 4

The difference between the value table obtained as a result of iteration 4 and iteration 3 is very small. So, we can say that we have attained convergence and we take the value table obtained as a result of iteration 4 as our optimal value function.

Step 2 – Extract the optimal policy from the optimal value function obtained from step 1

With the latest value table, compute Q values for all actions in all state,

The Q value for action 0 in state **A** is computed as:

$$\begin{aligned} Q(A,0) &= P^0_{AA}[R^0_{AA} + \gamma V(A)] + P^0_{AB}[R^0_{AB} + \gamma V(B)] + P^0_{AC}[R^0_{AC} + \gamma V(C)] \\ &= 0.1(0 + 1.95) + 0.8(-1 - 0.72) + 0.1(1 + 1.3) \\ &= -0.951 \end{aligned}$$

The Q value for action 1 in state **A** is computed as:

$$\begin{aligned} Q(A,1) &= P^1_{AA}[R^1_{AA} + \gamma V(A)] + P^1_{AB}[R^1_{AB} + \gamma V(B)] + P^1_{AC}[R^1_{AC} + \gamma V(C)] \\ &= 0.1(0 + 1.95) + 0.0(-1 - 0.72) + 0.9(1 + 1.3) \\ &= 2.26 \end{aligned}$$

Similarly compute Q(B,0),Q(B,1),Q(C,0),Q(C,1) and Q table becomes as follows,

State	Action	Value
A	0	- 0.95
A	1	2.26
B	0	- 0.5
B	1	0.5
C	0	- 1.1
C	1	1.4

Q table

We can extract the optimal policy by selecting the action that has the maximum Q value.

$$\pi^* = \arg\max_a Q(s,a)$$

- From this Q table, we pick the action in each state that has the maximum value as an optimal policy. Thus, our optimal policy would select action 1 in state A, action 1 in state B, and action 1 in state C.
- Thus, according to our optimal policy, if we perform action 1 in state A, we can reach state C without visiting state B.