Reinforcement Learning

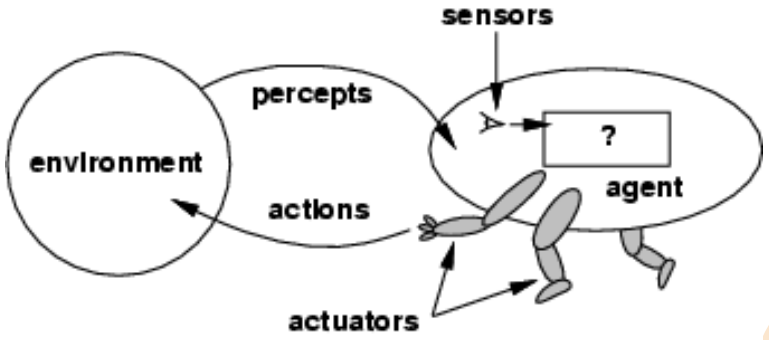
DR. BHARGAVI R

SCOPE

VIT CHENNAI

Introduction

- Learning happens from interaction of the Agent with the environment to achieve some long-term goal that is related to the state of the environment.

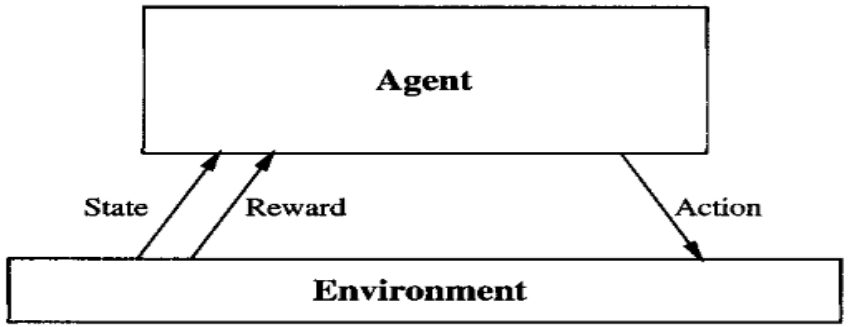


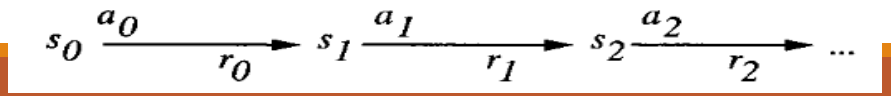
- Examples:
  - learning to control a mobile robot.
  - learning to optimize operations in factories.
  - learning to play board games etc,.
- How to learn successful control policies by experimenting in their environment

Introduction (cont…)

- Each time the agent performs an action in its environment, a trainer may provide a reward or penalty to indicate the desirability of the resulting state.
- Example: when training an agent to play a game the trainer might provide a positive reward when the game is won, negative reward when it is lost, and zero reward in all other states.
- The task of the agent is to learn from this indirect, delayed reward, to choose sequences of actions that produce the greatest cumulative reward.
- Goal: Choose actions that maximize cumulative reward

$r + \gamma r + \gamma^2 r + \ldots$, where $0 <= \gamma < 1$





How Reinforcement Learning is Different from (Un)Supervised Learning

- Target function to be learned - Control policy that outputs an appropriate action a from the set A, given the current state s from the set S. $\pi : S \rightarrow A$

Delayed reward:

- The task of the agent is to learn a target function π that maps from the current state s to the optimal action $a = \pi(s)$.
- Training example is not of the form $< s, \pi(s) >$, instead the trainer provides only a sequence of immediate reward values as the agent executes its sequence of actions.

Exploration vs Exploitation:

- The learner faces a tradeoff in choosing whether to favor ***exploration*** of unknown states and actions (to gather new information), or ***exploitation*** of states and actions that it has already learned will yield high reward

How RL is Different ….

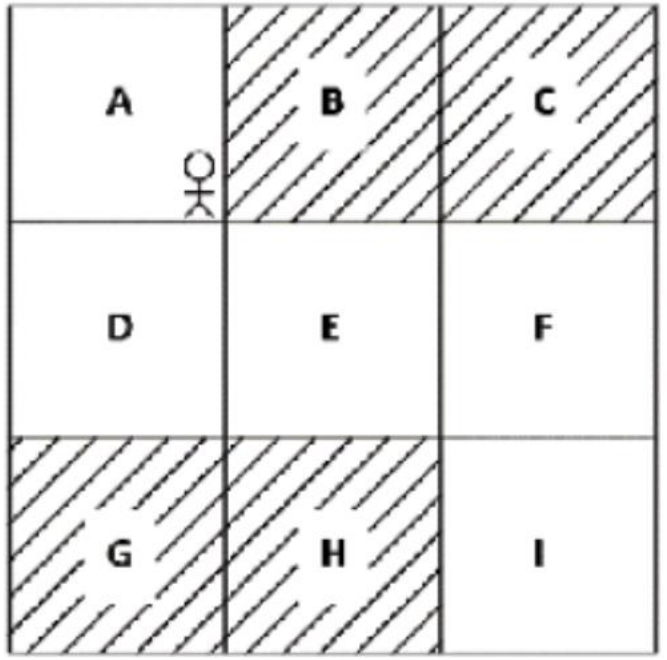
Partially observable states:

- In many practical situations sensors provide only partial information.
- In such cases, it may be necessary for the agent to consider its previous observations together with its current sensor data when choosing actions, and the best policy may be one that chooses actions specifically to improve the observability of the environment

Life-long learning:

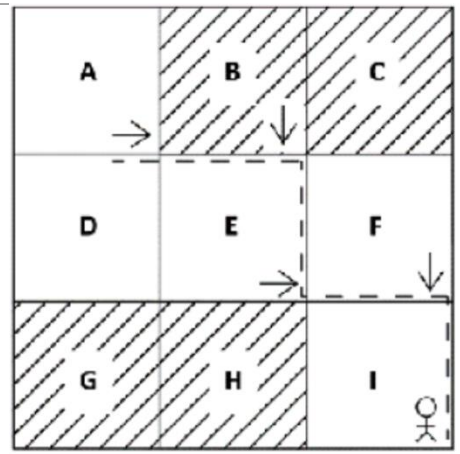
- Robot learning often requires that the robot learn several related tasks within the same environment, using the same sensors.

Grid World Problem – Reinforcement Learning

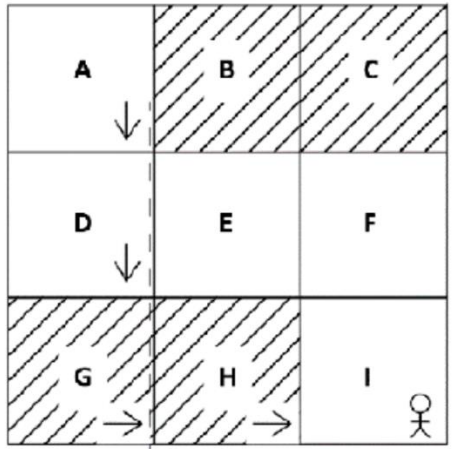


Grid World Problem (solution)

Iteration 1

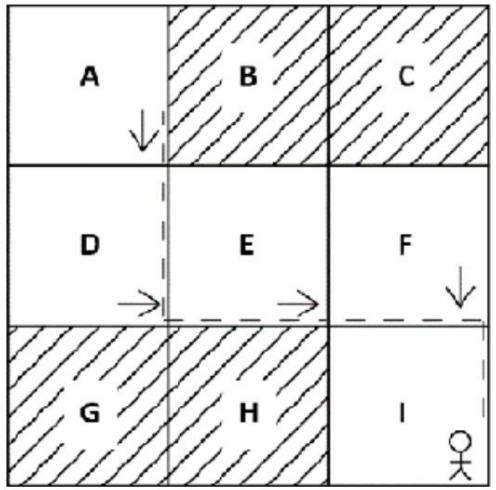


Iteration 2



Grid World Problem (solution)

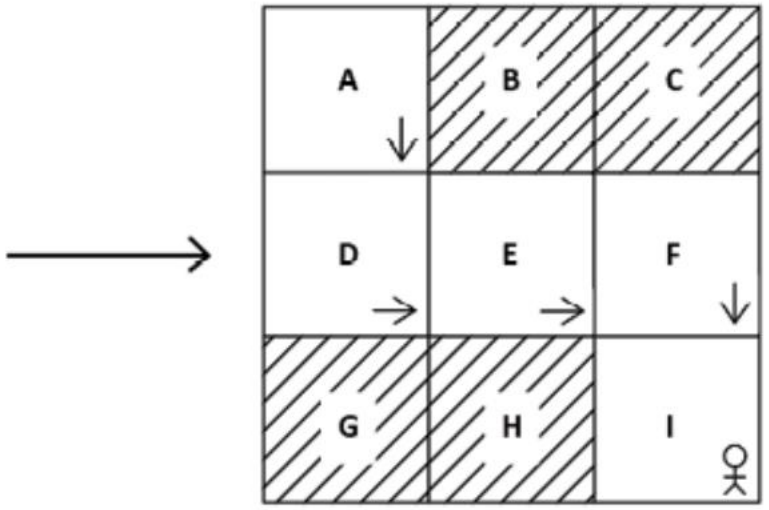
Iteration 3



Grid World Problem (solution)

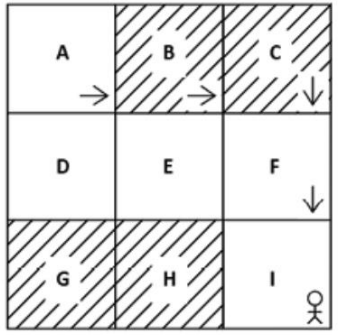
Optimal Policy

State	Action
A	Down
D	Right
E	Right
F	Down



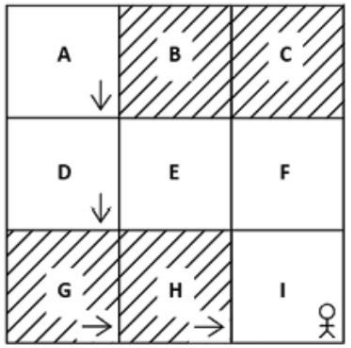
Episode

Episode 1



$\tau_1 =$ A —Right, -1→ B —Right, -1→ C —Down, +1→ F —Down, +1→ I

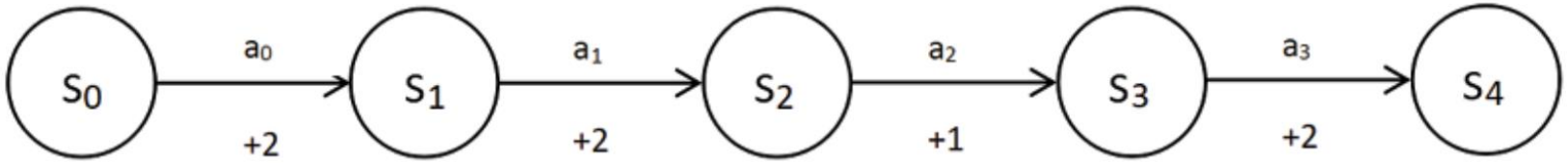
Episode 2



$\tau_2 =$ A —Down, +1→ D —Down, -1→ G —Right, -1→ H —Right, +1→ I

Return and Discount Factor

The goal of an agent is to maximize the return, that is, maximize the sum of rewards (cumulative rewards) obtained over the episode.



Cumulative return $R(\mathcal{T}) = r_0 + r_{1+} r_{2+} \text{-------} + r_T = 2 + 2 + 1 + 2 = 7$

Discounted Rewards : $r + \gamma r + \gamma^2 r + \ldots$, where $0 <= \gamma < 1$; γ is called Discount factor.

Small discount factor: More importance to the immediate reward than future rewards.

Large discount factor: more importance to future rewards than the immediate reward.

Return and Discount Factor (cont…)

- Why Discount Factor?
- To Ensure Convergence in Infinite Horizon Problems: In tasks that don't have a natural end (continuous tasks), the sum of future rewards could potentially be infinite if we don't discount them.
- By multiplying future rewards by a discount factor, we create a geometric series that converges to a finite value, making the learning problem mathematically tractable.
- There's uncertainty about the future – the environment might change, the agent might fail, or other unforeseen events could occur. Discounting gives more weight to immediate rewards, reflecting this preference for sooner rather than later.
- To Prevent Oscillations and Promote Stability.

RL – Approaches

Value Based

In value-based RL, the goal is to optimize the value function V(s).

The value function is a function that tells us the maximum expected future reward the agent will get at each state.

The value of each state is the total amount of the reward an agent can expect to accumulate over the future, starting at that state.

The agent will use the value function to select which state to choose at each step. The agent takes the state with the biggest value.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

RL – Approaches

Policy Based

In policy-based RL, the goal is to directly optimize the policy function $\pi(s)$ without using a value function.

We learn a policy function. This lets us map each state to the best corresponding action.

- Deterministic: a policy at a given state will always return the same action.
- Stochastic: output a distribution probability over actions.

Model Based

In model-based RL, we model the environment i.e we create a model of the behavior of the environment.

The problem is each environment will need a different model representation.

Formalization of Learning Task

Formulation of the problem of learning sequential control strategies –

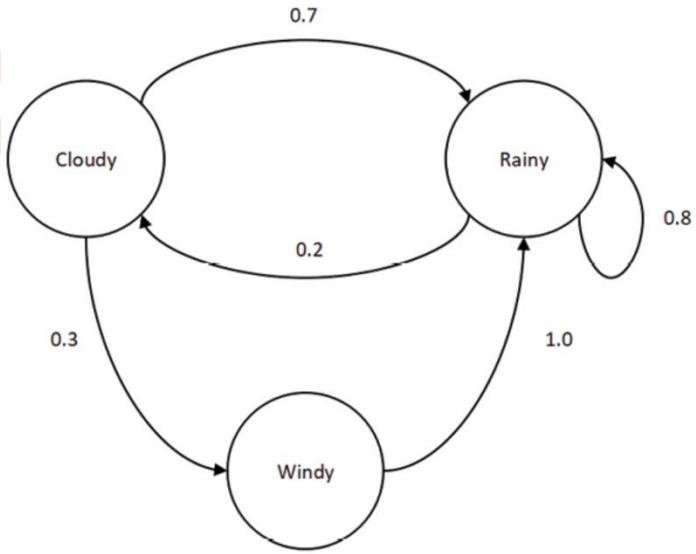
- Based on Markov Decision Process(MDP)
- Set of states S (Assumption - Finite set)
- Set of actions A (Assumption - Finite set)
- At each discrete time step t, the agent senses the current state s_t, chooses a current action a_t, and performs it.
- The environment responds by giving the agent a reward $r_t = r(s_t, a_t)$ and by producing the succeeding state $s_{t+1} = \delta(s_t, a_t)$
- The functions r and δ (Assumption - r and δ are deterministic) are not known to the agent and they depend only on the current state and action, and not on earlier states or actions.

State Transition & Transition Probability

State Transition: Moving from one state to another is called a transition

Transition probability: Probability of transition from one state to another state is called a transition probability and it is denoted by p(s| s`)

Current State	Next State	Transition Probability
Cloudy	Rainy	0.7
Cloudy	Windy	0.3
Rainy	Rainy	0.8
Rainy	Cloudy	0.2
Windy	Rainy	1.0



	Cloudy	Rainy	Windy
Cloudy	0.0	0.7	0.3
Rainy	0.2	0.8	0.0
Windy	0.0	1.0	0.0

Policy

- A **policy** is a fundamental concept that defines the agent's behavior.
- It essentially maps states of the environment to actions that the agent should take when in those states.
- In simpler terms, it is the agent's strategy for interacting with the environment.
- **Deterministic Policy**: For each state, the policy specifies a single, definite action that the agent will take. This can be represented as a function: $a = \pi(s)$
- **Stochastic Policy**: For each state, the policy specifies a probability distribution over the possible actions.
- This means that the agent might take different actions in the same state with certain probabilities. This can be represented as: $\pi(a|s) = P(A_t = a \mid S_t = s)$

Policy (cont…)

Deterministic policy

Maps states ⟶ Action

Example :

A ⟶ Down

Stochastic policy

Maps states ⟶ Probability distribution over action space

Example :

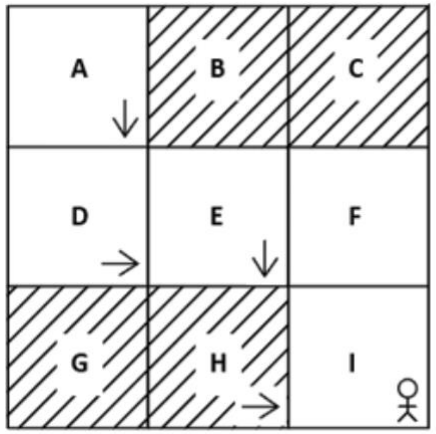
A ⟶ [0.10, 0.70, 0.10, 0.10]
up down left right

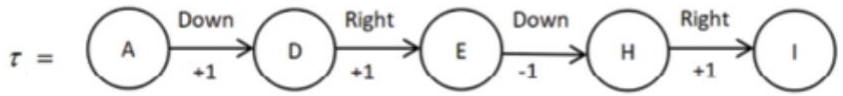
Value Function

- Also called the state value function, denotes the value of the state.
- The value of a state is the return an agent would obtain starting from that state following policy π .
- The value of a state or value function is usually denoted by V(s) and it can be expressed as:

$$V^{\pi}(s) = [R(\tau)|s_0 = s]$$

Value Function - Example





The value of state **A** is the return of the trajectory starting from state **A** $V(A) = 1+1+ -1+1 = 2$.

The value of state **D** is the return of the trajectory starting from state **D**. $V(D) = 1-1+1= 1$.

The value of state **E** is the return of the trajectory starting from state **E**. $V(E) = -1+1 = 0$.

The value of state H is the return of the trajectory starting from state H. Thus, V(H) = 1.

Value Function (cont…)

Since return is the random variable and it takes different values with some probability, the value function of state s can be defined as the expected return that the agent would obtain starting from state s following policy π.

It can be written as $$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

suppose we are in state A and the stochastic policy returns the probability distribution over the action space as [0.0,0.80,0.00,0.20].

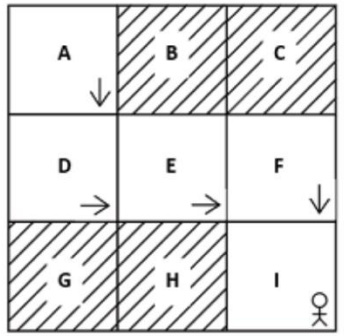
It implies that with the stochastic policy, in state A, we perform the action down 80% of the time, and the action right 20% of the time.

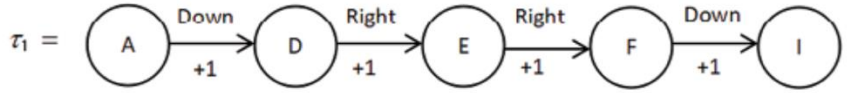
Also, say our stochastic policy selects the action right in states D and E and the action down in states B and F 100% of the time.

Value Function (cont…)

Episode 1 using stochastic policy π

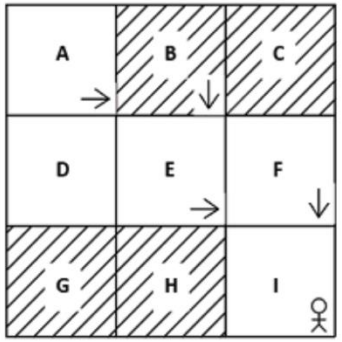
The value of state A, V(A) = 1 + 1+ 1+1 = 4

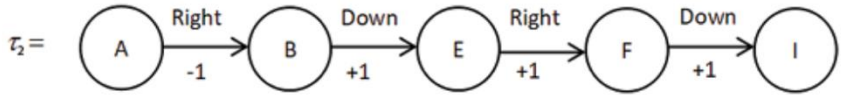




Now consider another episode 2

The value of state A, V(A) = -1+1+1+1 = 3





The expected return is the weighted average, that is, the sum of the return multiplied by their probability.

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

Value Function (cont…)

The Value of state A can be computed as

$$V^{\pi}(A) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = A]$$

$$= \sum_i R(\tau_i)\pi(a_i|A)$$

$$= R(\tau_1)\pi(\text{down}|A) + R(\tau_2)\pi(\text{right}|A)$$

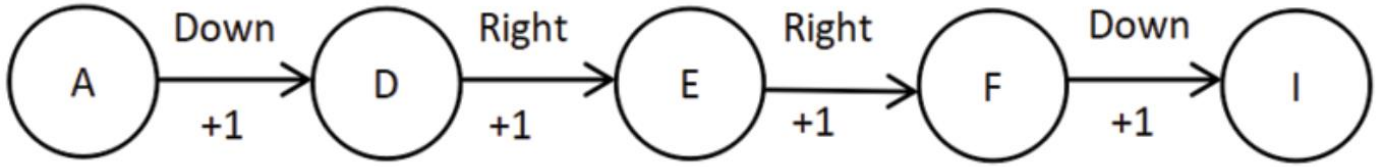
$$= 4\ (0.8) + 3\ (0.2) = 3.8$$

Q Function

- A Q function, also called the state-action value function, denotes the value of a state-action pair.
- The value of a state-action pair is the return the agent would obtain starting from state s and performing action a following policy π.
- The value of a state-action pair or Q function is usually denoted by Q(s,a) and is known as the Q value or state-action value. It is expressed as:

$$Q^{\pi}(s,a) = [R(\tau)|s_0 = s, a_0 = a]$$

Q Function - Example



Q value of state-action pair A-down: The return of our trajectory starting from state A and performing the action ***down***:

$$Q^{\pi}(A, \text{down}) = [R(\tau)|s_0 = A, a_0 = \text{down}]$$

$$Q(A, \text{down}) = 1 + 1 + 1 + 1 = 4$$

Similarly Q value of state-action pair D-***Right***: The return of our trajectory starting from state D and performing the action ***Right***: 1 + 1 + 1 = 3

$$Q^{\pi}(s, a) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau)|s_0 = s, a_0 = a]$$

Discounted Cumulative Rewards

- The task of the agent is to learn a policy $\pi : S \rightarrow A$, for selecting its next action a_t based on the current observed state s_t; that is, $\pi(s_t) = a_t$
- Which policy? - Policy that produces the greatest possible cumulative reward for the robot over time.
- we define the cumulative value $V^{\pi}(s_t)$ achieved by following an arbitrary policy π from an arbitrary initial state s_t as follows:

$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \qquad 0 \leq \gamma < 1$$

- $V^{\pi}(s)$ is called as Discounted cumulative reward achieved by policy π from initial state s

Optimal Policy

- The constant γ determines the relative value of delayed versus immediate rewards.
- If $\gamma = 0$, only the immediate reward is considered and if γ closer to 1, future rewards are given greater emphasis relative to the immediate reward.
- We require that the agent learn a policy π that maximizes $V^{\pi}(s)$ for all states s. We will call such a policy an optimal policy and denote it by π^*.

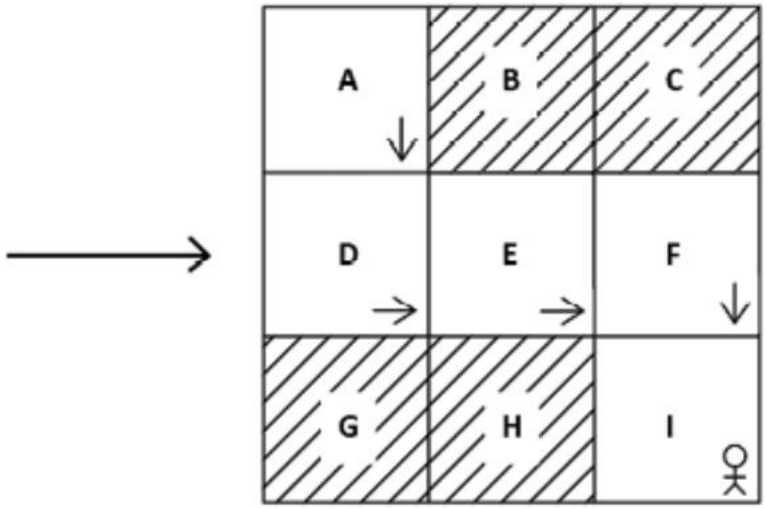
$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$

- $V^*(s)$ is the value function of the optimal policy, and it gives the maximum discounted cumulative reward that the agent can obtain starting from state s

Example

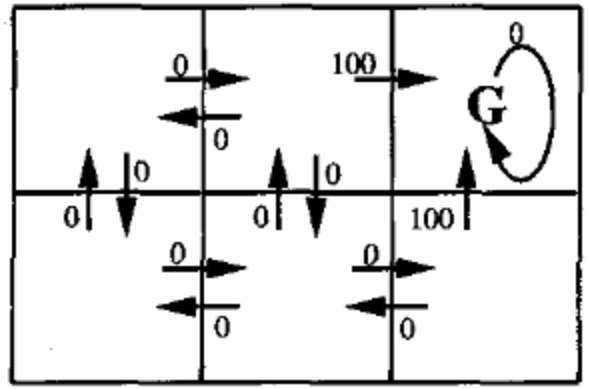
Optimal Policy

State	Action
A	Down
D	Right
E	Right
F	Down

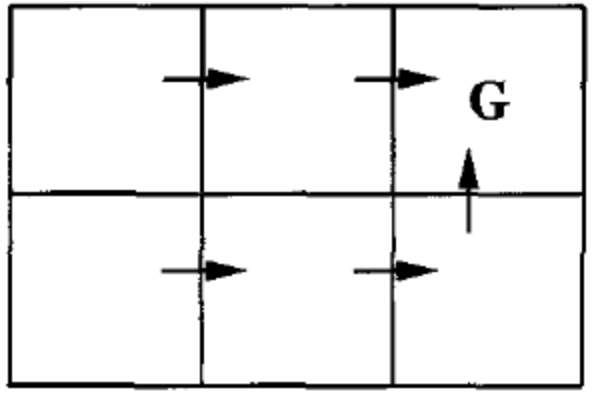


Example

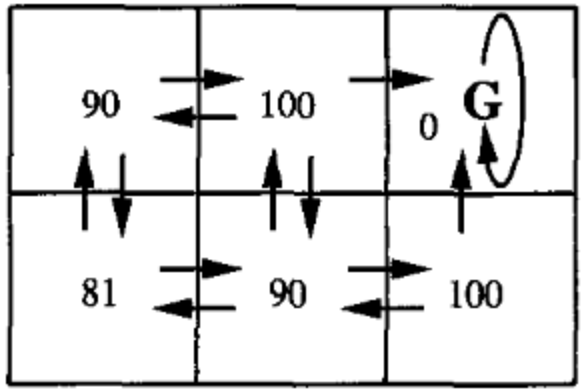
Assume Gamma = 0.9



$r(s, a)$ (immediate reward) values



One optimal policy



$V^*(s)$ values

Q Table

Consider a system with two just two states S_0 and S_1.

Let 0, and 1 be two actions that are possible in each of these states.

Then Q table can be represented as shown here.

State	Action	Value
s_0	0	9
s_0	1	11
s_1	0	17
s_1	1	13

Q	Action - 0	Action - 1
S_0	9	11
S_1	17	13

Q Learning

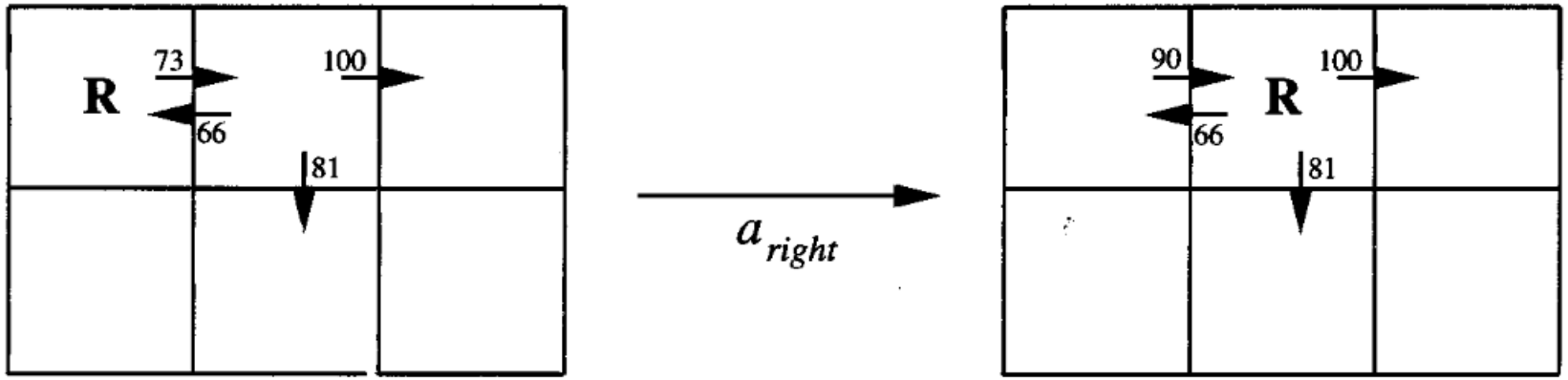
For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state s
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Example – Q Value updation



$$\hat{Q}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$

$$\leftarrow 0 + 0.9 \ \max\{66, 81, 100\}$$

$$\leftarrow 90$$

Example

Imagine an agent in a 3x1 grid world with three states: **S1, S2, and S3**. The agent starts in **S1**, and its goal is to reach **S3**, where it receives a reward. The environment works as follows:

States: S1, S2, S3

Actions: Move Right, Move Left

Rewards:

- If the agent reaches S3, it gets a reward of **+1**.
- All other moves give a reward of **0**.

Goal: The agent needs to learn the Q-values so that it can maximize its reward, eventually learning to move from S1 to S3.

Discount factor (γ) = 0.9

Example (cont…)

Assume the Q-values for each action in each state are initialized to **0**.

Q(S1,Move Right)= 0,Q(S1,Move Left)= 0

Q(S2,Move Right)= 0,Q(S2,Move Left)= 0

Q(S3,Move Right)=0,Q(S3,Move Left)= 0

Episode 1:

State = S1, Action = Move Right

- Agent moves to **S2**.
- **Reward** = 0 (no reward for this transition).
- **Next State = S2**

Example (cont…)

Q-Update for Q(S1, Move Right):

- Using the Q-learning update rule:

  Q(s1, Right) = 0 + 0.9(max(0,0)) = 0

  New sate = s2

State = S2, Action = Move Right

- Agent moves to **S3**.
- **Reward** = 1. **Next State = S3**

Q-Update for Q(S2, Move Right):

- Using the Q-learning update rule: Q(s2, Right) = 1 + 0.9(max(0,0)) = 1

New sate = s3

epsilon-greedy strategy

- The epsilon-greedy strategy is a simple yet effective method used in Q-learning (and other Reinforcement Learning algorithms) to balance the exploration-exploitation trade-off.
- At each step in the learning process, when the agent needs to choose an action in a given state, the epsilon-greedy strategy dictates the following:
- Exploration (with probability ε): With a small probability (ε, where $0 \leq \varepsilon \leq 1$), the agent chooses an action randomly from the set of possible actions available in the current state. This allows the agent to explore the environment and potentially discover better actions that it hasn't tried much or doesn't yet know have high Q-values.
- Exploitation (with probability 1 - ε): With a high probability (typically), the agent chooses the action that it currently believes to be the best based on its learned Q-values. This means it selects the action a that maximizes Q(s, a) for the current state s. The agent is exploiting its current knowledge to get the highest immediate reward.

Q learning with TD

Update rule for Q:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

SARSA (**State-Action-Reward-State- Action)** : It is an on-policy Reinforcement learning algorithm used to learn a policy for an agent to maximize cumulative rewards.

Unlike Q-learning, which is off-policy, SARSA updates the Q-value based on the actions chosen by the agent's current policy, making it an on-policy algorithm.

Update rule for Q :

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$