Reinforcement Learning
(Passive Learning- Direct Utility Estimation)

Dr. Virendra Singh Kushwah

Assistant Professor Grade-II

School of Computing Science and Engineering

Virendra.Kushwah@vitbhopal.ac.in

7415869616

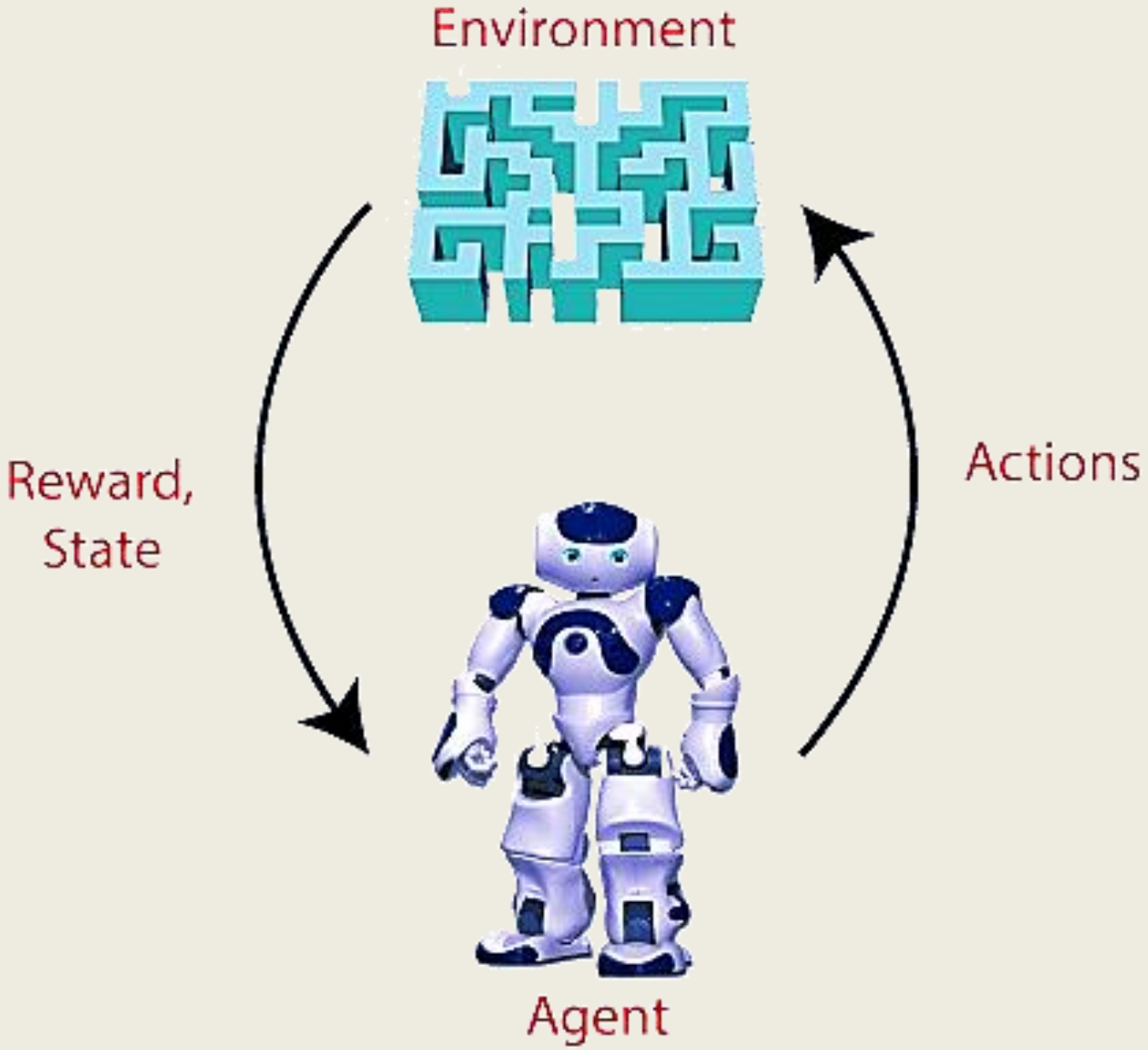
What is Reinforcement Learning?

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.

- ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."***

- How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.

- Example: Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (take action, change state/remain in the same state, and get feedback), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.


Environment
Reward,
State
Actions
Agent

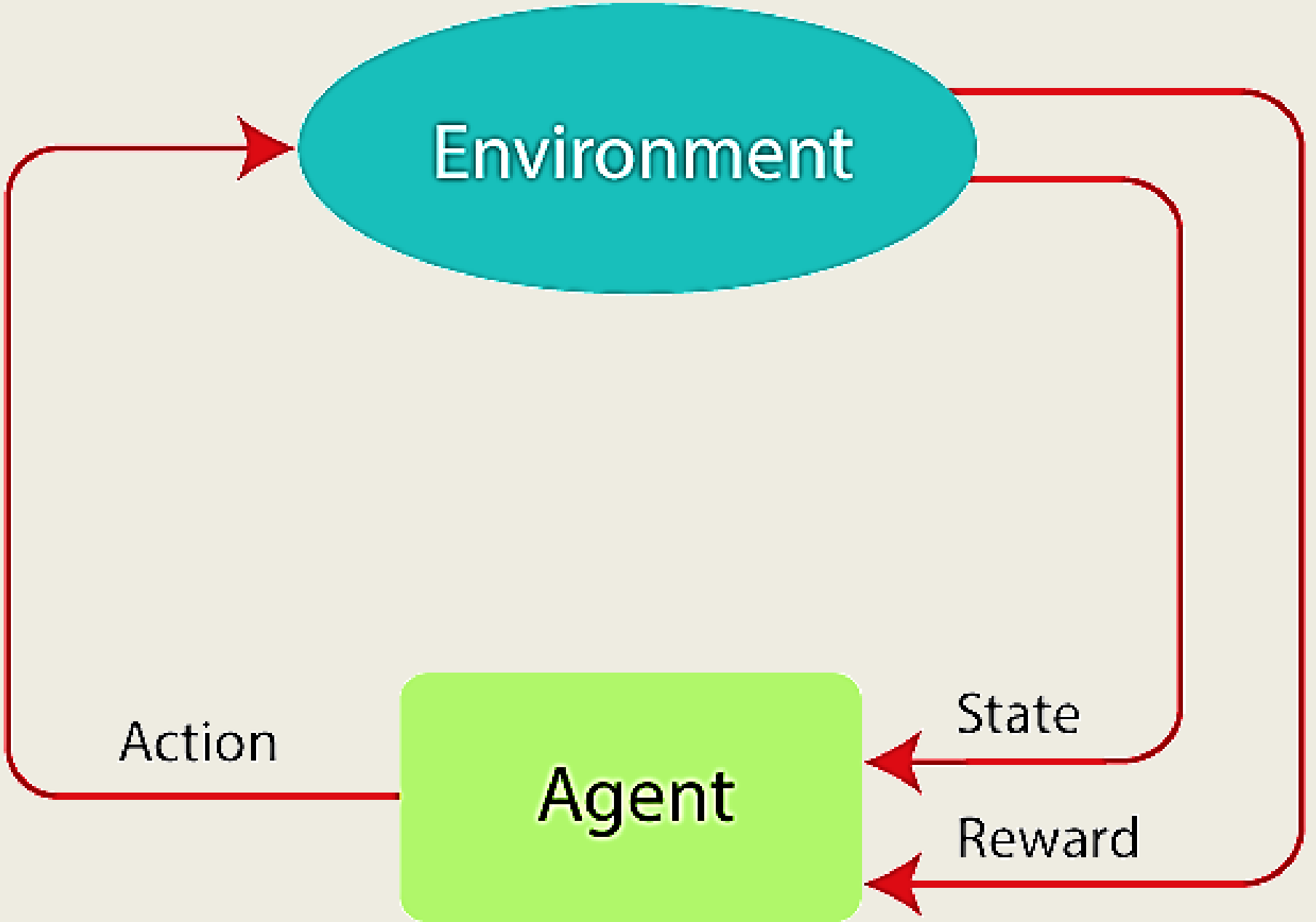
Terms used in Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Markov Decision Process

- Markov Decision Process or MDP, is used to formalize the reinforcement learning problems. If the environment is completely observable, then its dynamic can be modeled as a Markov Process.

- In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.


Environment
Action
Agent
State
Reward

- MDP is used to describe the environment for the RL, and almost all the RL problem can be formalized using MDP.

- MDP contains a tuple of four elements (S, A, Pa, Ra):

  - A set of finite States S
  - A set of finite Actions A
  - Rewards received after transitioning from state S to state S', due to action a.
  - Probability Pa.

- MDP uses Markov property, and to better understand the MDP, we need to learn about it.

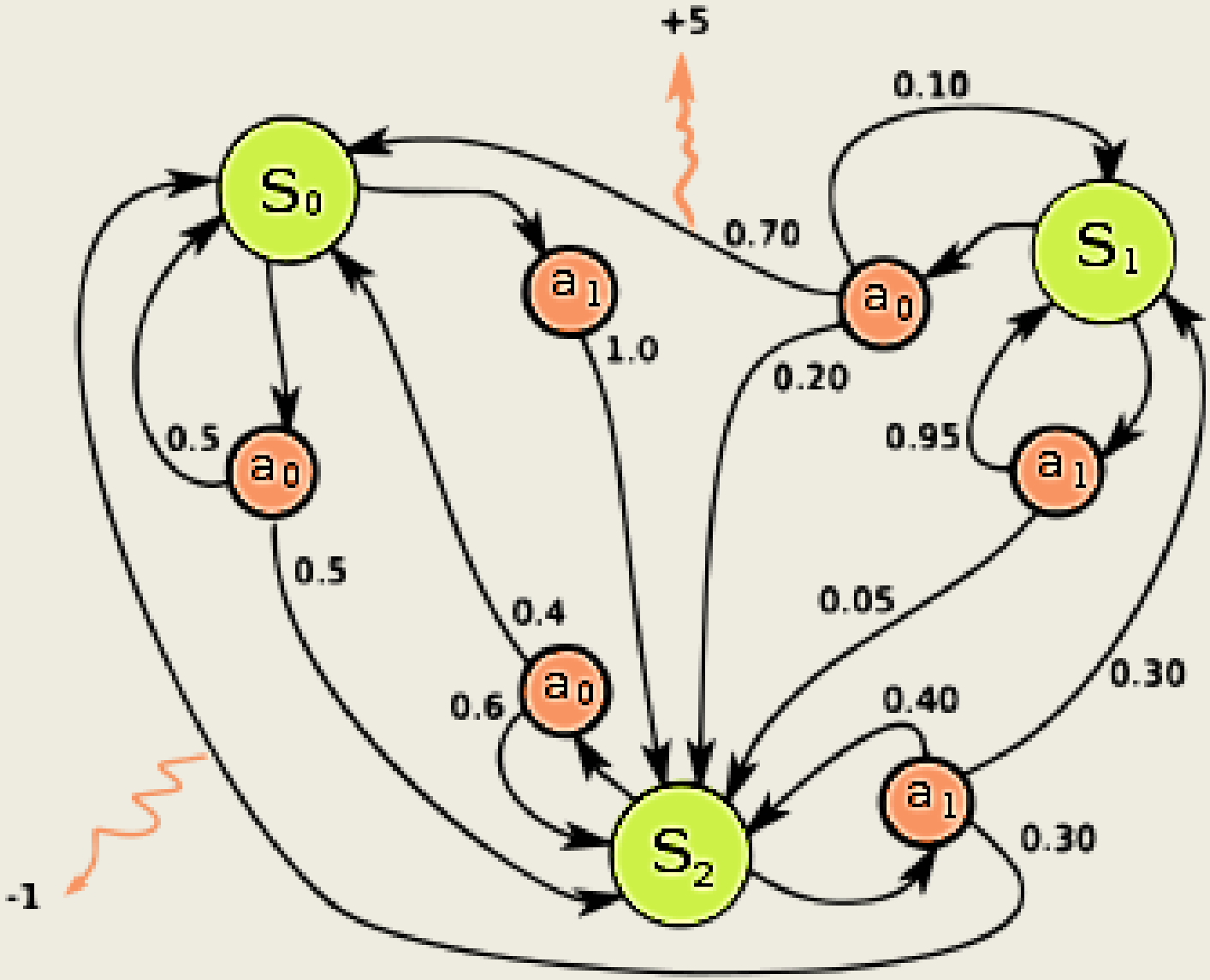
- Markov Property:
- It says that "If the agent is present in the current state S1, performs an action a1 and move to the state s2, then the state transition from s1 to s2 only depends on the current state and future action and states do not depend on past actions, rewards, or states."
- Or, in other words, as per Markov Property, the current state transition does not depend on any past action or state. Hence, MDP is an RL problem that satisfies the Markov property. Such as in a Chess game, the players only focus on the current state and do not need to remember past actions or states.

- Finite MDP:
- A finite MDP is when there are finite states, finite rewards, and finite actions. In RL, we consider only the finite MDP.

- Markov Process:
- Markov Process is a memoryless process with a sequence of random states S1, S2, ..., St that uses the Markov Property. Markov process is also known as Markov chain, which is a tuple (S, P) on state S and transition function P. These two components (S and P) can define the dynamics of the system.

What are required elements to solve an RL problem?

- Let's consider a problem where the agent can be in various states and can choose an action from a set of actions. Such type of problems are called Sequential Decision Problems.
- An MDP captures a fully observable, non-deterministic environment with Markovian Transition Model and additive rewards in which such an agent acts. The solution to an MDP is an optimal policy which refers to the choice of action for every state that maximizes overall cumulative reward. Thus, the transition model that represents an agent's environment(when the environment is known) and the optimal policy which decides what action the agent needs to perform in each state are required elements for training the agent learn a specific behavior.



What is meant by passive and active reinforcement learning and how do we compare the two?

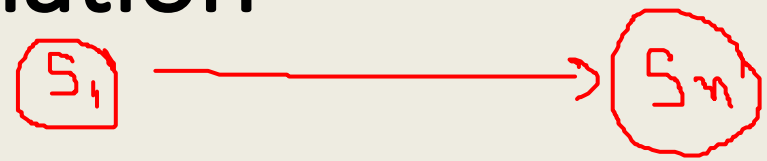
- Both active and passive reinforcement learning are types of RL.
- In case of passive RL, the agent's policy is fixed which means that it is told what to do.
- In contrast to this, in active RL, an agent needs to decide what to do as there's no fixed policy that it can act on.
- Therefore, the goal of a passive RL agent is to execute a fixed policy (sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.

What are some common active and passive RL techniques?

Passive Learning

- As the goal of the agent is to evaluate how good an optimal policy is, the agent needs to learn the expected utility $U^{\pi}(s)$ for each state *s*. This can be done in three ways.

Direct Utility Estimation



- In this method, the agent executes a sequence of trials or runs (sequences of states-actions transitions that continue until the agent reaches the terminal state). Each trial gives a sample value and the agent estimates the utility based on the samples values. Can be calculated as running averages of sample values. The main drawback is that this method makes a wrong assumption that state utilities are independent while in reality they are Markovian. Also, it is slow to converge.
- Suppose we have a 4x3 grid as the environment in which the agent can move either Left, Right, Up or Down(set of available actions). An example of a run,

Suppose we observe the following trial:

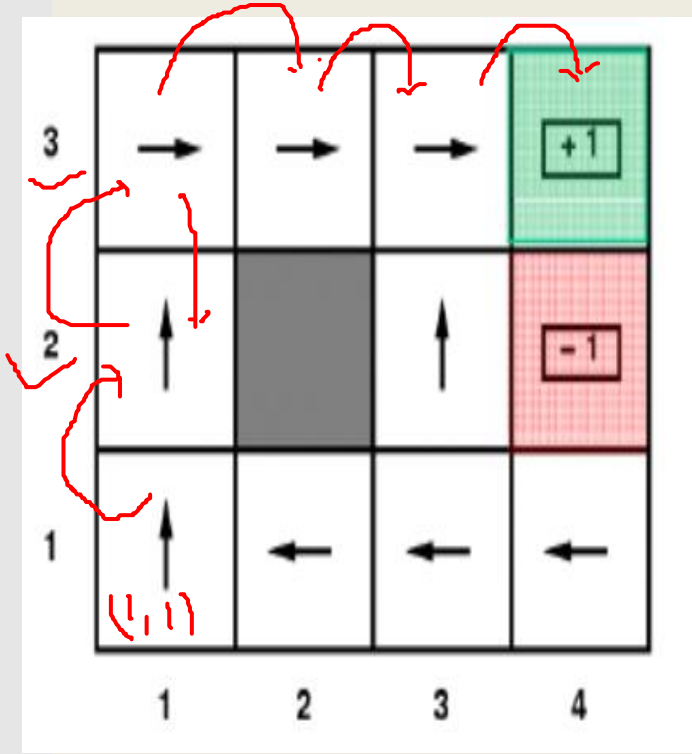
$(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04}$
$\rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$

The total reward starting at (1,1) is 0.72. We call this a sample of the observed-reward-to-go for (1,1).

For (1,2) there are two samples for the observed-reward-to-go (assuming $\gamma=1$):

1. $(1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$ [Total: 0.76]

2. $(1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$ [Total: 0.84]



4x3

- Direct Utility Estimation keeps a running average of the observed reward-to-go for each state
- E.g. For state (1,2), it stores (0.76+0.84)/2 = 0.8
- As the number of trials goes to infinity, the sample average converges to the true utility

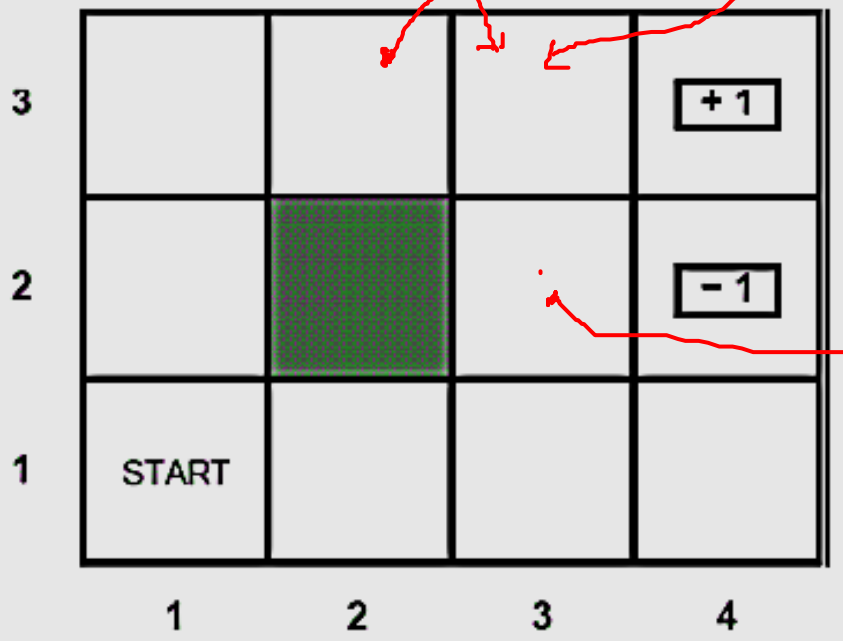
- The big problem with Direct Utility
- Estimation: it converges very slowly!
- Why?
- – Doesn't exploit the fact that utilities of states are
- not independent
- – Utilities follow the Bellman equation

$$U_\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U_\pi(s')$$

Note the dependence on neighboring states

Bellman equation

Using the dependence to your advantage:



Suppose you know that state (3,3) has a high utility

Suppose you are now at (3,2)

The Bellman equation would be able to tell you that (3,2) is likely to have a high utility because (3,3) is a neighbor.

DEU can't tell you that until the end of the trial

Remember that each blank state has R(s) = -0.04