VIT BHOPAL
www.vitbhopal.ac.in

Reinforcement Learning

(Active Reinforcement learning –Exploration, Q-Learning)

Dr. Virendra Singh Kushwah

Assistant Professor Grade-II

School of Computing Science and Engineering

Virendra.Kushwah@vitbhopal.ac.in

7415869616

Active Reinforcement Learning

- So far, we've assumed agent has a policy
  - We just learned how good it is

- Now, suppose agent must learn a good policy (ideally optimal)
  - While acting in uncertain world


lsnes rr2-β23 [bsnes v085 (Compatibility core)]
File SNES Movie Speed Tools Configure Help
MARIO
× 5
× 0
TIME
368
0×
0
0
Frame: 4765 Lag: 443 Subframe: 0 Slot: 1 [Nonexistent] Speed: 74% Recording


x o What the model sees (White area is focused by spatial transformer)
AR+B
AU+X
AR+B


x o Recieved direct reward and discounted expected future reward (gamma=0.90)
Recieved direct reward and discounted expected future reward (gamma=0.90)
10
0
-10
-20
-30
4900
4920
4940
4960
4980
State

ADP with exploration function

- As the goal of an active agent is to learn an optimal policy, the agent needs to learn the expected utility of each state and update its policy. Can be done using a passive ADP agent and then using value or policy iteration it can learn optimal actions. But this approach results into a greedy agent. Hence, we use an approach that gives higher weights to unexplored actions and lower weights to actions with lower utilities.

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} f\left(\sum_{s'} P(s'|s,a) U_i(s'), N(s,a)\right)$$

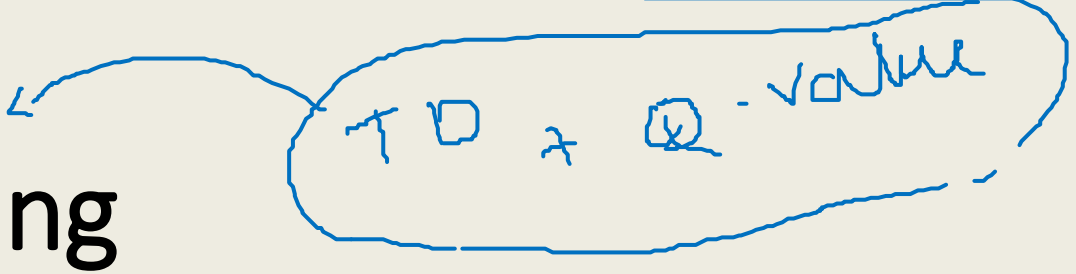
A classical approach to any **reinforcement learning** (RL) problem is to **explore** and to exploit. **Explore** the most rewarding way that reaches the target and keep on exploiting a certain action; **exploration** is hard. Without proper reward functions, the algorithms can end up chasing their own tails to eternity.

- Where f(u, n) is the exploration function that increases with expected value u and decreases with number of tries n

$$f(u,n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases}$$

R+ is an optimistic reward and *Ne* is the number of times we want an agent to be forced to pick an action in every state. ***The exploration function converts a passive agent into an active one.***

Q-Learning



- Q-learning is a TD learning method which does not require the agent to learn the transitional model, instead learns Q-value functions Q(s, a) .

$$U(s) = max_a\, Q(s,a)$$

Q-values can be updated using the following equation,

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

- Next action can be selected using the following policy,

$$a_{next} = arg\ max_{a'} f(Q(s',a'), N(s',a'))$$

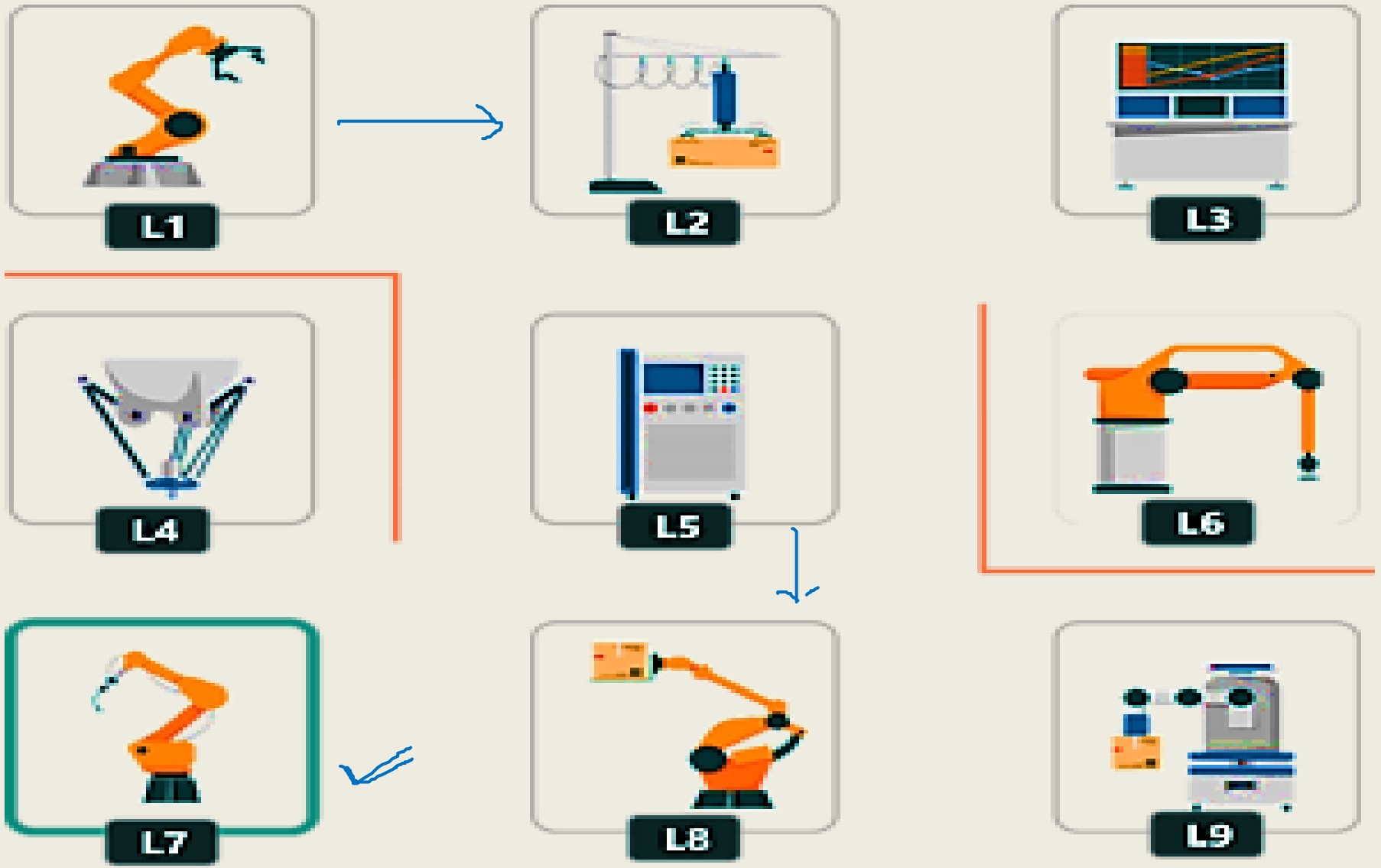
Again, this is simpler to compute but slower than ADP.

- Let’s understand what is Q learning with our problem statement here. It will help us to define the main components of a reinforcement learning solution i.e. agents, environment, actions, rewards, and states.

Automobile Factory Analogy:

- We are at an Automobile Factory filled with robots. These robots help the Factory workers by conveying the necessary parts required to assemble a car.
- These different parts are located at different locations within the factory in 9 stations. The parts include Chassis, Wheels, Dashboard, Engine and so on. Factory Master has prioritized the location where chassis is being installed as the highest priority.

Let’s have a look at the setup here:

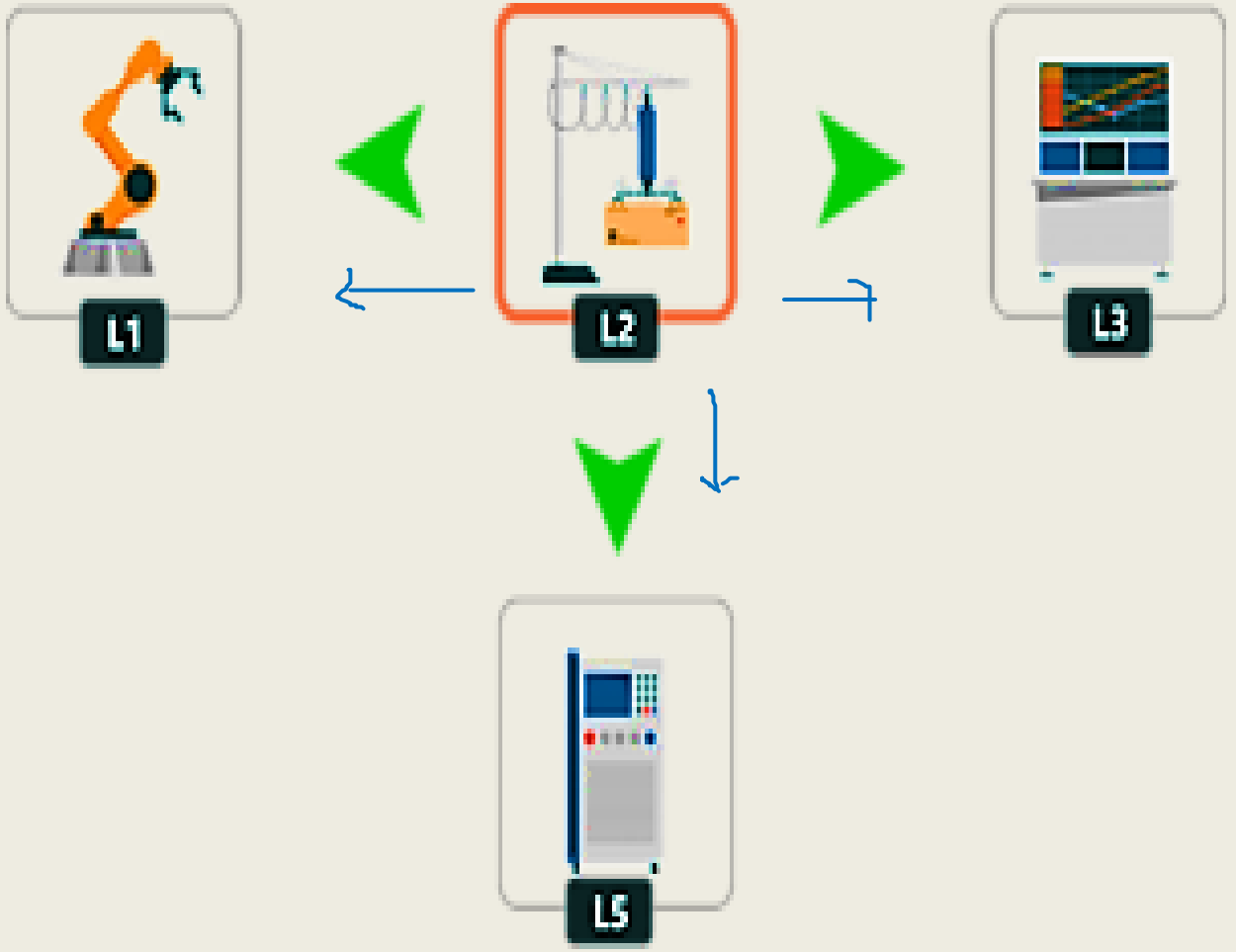


States:

The Location at which a robot is present at a particular instance is called its state. Since it is easy to code it rather than remembering it by names. Let's map the location to numbers.

Actions:

- Actions are nothing but the moves made by the robots to any location.
- Consider, a robot is at the L2 location and the direct locations to which it can move are L5, L1 and L3. Let's understand this better if we visualize this:



Rewards:

- A Reward will be given to the robot for going directly from one state to another. For example, you can reach L5 directly from L2 and vice versa. So, a reward of 1 will be provided in either case. Let's have a look at the reward table:

	L1	L2	L3	L4	L5	L6	L7	L8	L9
L1	0	1	0	0	0	0	0	0	0
L2	1	0	1	0	0	0	0	0	0
L3	0	1	0	0	0	1	0	0	0
L4	0	0	0	0	0	0	1	0	0
L5	0	1	0	0	0	0	0	1	0
L6	0	0	1	0	0	0	0	0	0
L7	0	0	0	1	0	0	0	1	0
L8	0	0	0	0	1	0	1	0	1
L9	0	0	0	0	0	0	0	1	0

- Remember when the Factory Master Prioritized the chassis location. It was L7, so we are going to incorporate this fact into our rewards table. So, we'll assign a very large number(999 in our case) in the (L7, L7) location.

	L1	L2	L3	L4	L5	L6	L7	L8	L9
L1	0	1	0	0	0	0	0	0	0
L2	1	0	1	0	0	0	0	0	0
L3	0	1	0	0	0	1	0	0	0
L4	0	0	0	0	0	0	1	0	0
L5	0	1	0	0	0	0	0	1	0
L6	0	0	1	0	0	0	0	0	0
L7	0	0	0	1	0	0	999	1	0
L8	0	0	0	0	1	0	1	0	1
L9	0	0	0	0	0	0	0	1	0

Model-free (real world)	Temporal Difference Learning (TD)	Q-learning
Model-based (simulation)	Adaptive Dynamic Programming(ADP)	ADP with proper exploration function

VIT®

BHOPAL

www.vitbhopal.ac.in