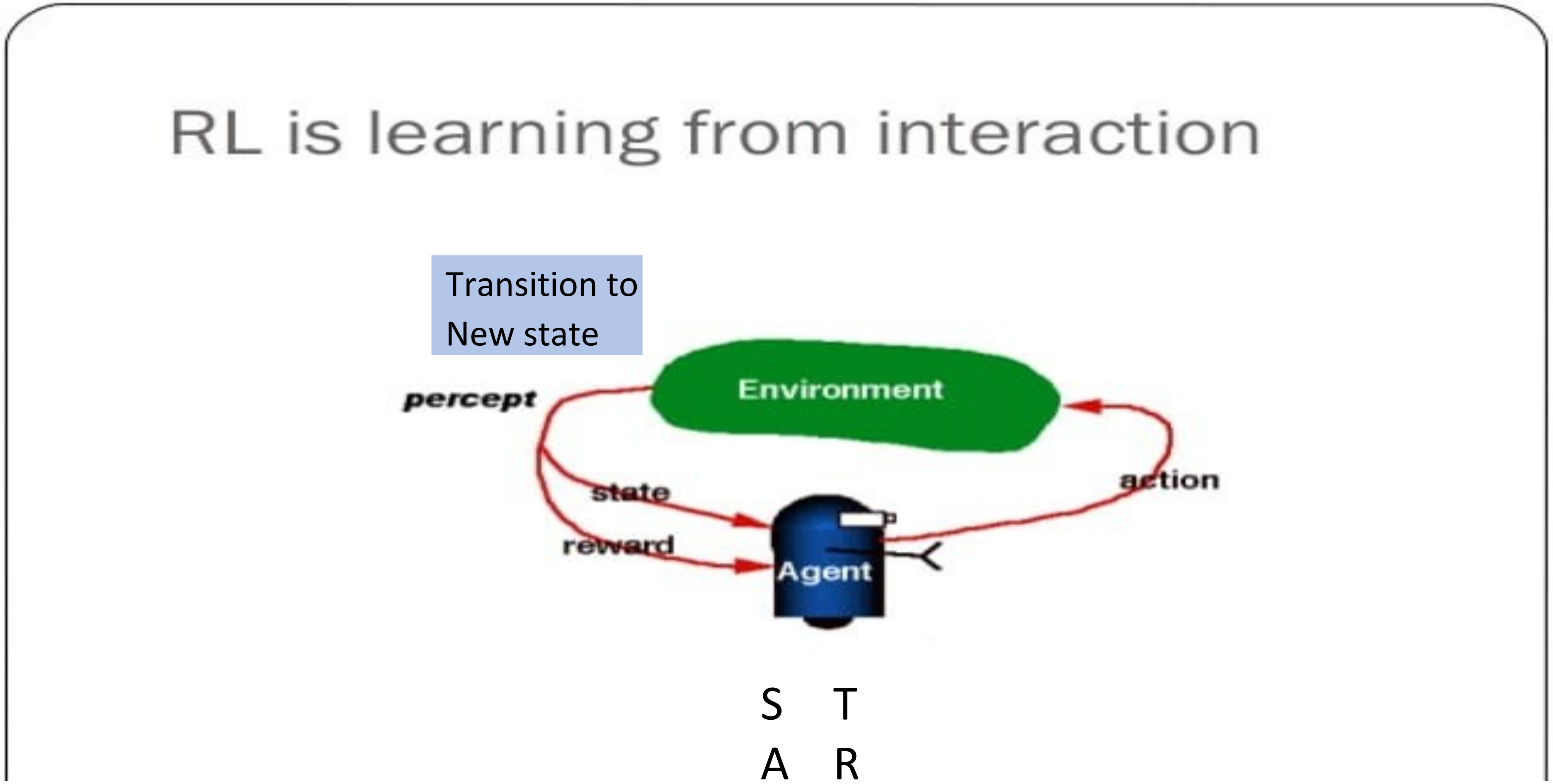
Reinforcement Learning

RL is learning from interaction



RL is learning from interaction

Transition to
New state

S T
A R

Markov Decision Process (MDP)

- Set of states S – actually features vector to describe environment & agent
- Set of actions A that can be taken at each state
- State transition probabilities p(s’ | s, a). Memory-less property assumed.
- Policy is mapping from States to possible actions P: S?A. This is the solution of MDP
- Finite MDP if both S and A are finite

Markov Decision Process (MDP)

- Rewards are a set of real numbers or discrete in S X A. Reward Functions:
    - State reward R(S)
    - Immediate reward R(S,a, S'),
    - R(S,a).
- Discount factor γ in [0, 1] – priority given to future experience. ~0.9
- Learning rate: α (~0.1)

Lodhi Garden visit – R: State Transition Immediate rewards r_{t+1}

$(S_t$ ⍰ $S_{t+1})$	**S_1 Tomb, hungry, tired**	**S_2 Garden, hungry, fresh**	**S_3 Lake, filled, tired**	**S_4 Eatery, filled, fresh**	**S_5 Gazebo, hungry, fresh**	**S_6 Out, filled, tired**
S_1 Tomb, hungry, tired	-10	50	70	100	50	0
S_2 Garden, hungry, fresh	20	40	100	100	20	0
S_3 Lake, filled, tired	0	70	20	0	0	50
S_4 Eatery, filled, fresh	100	0	0	0	0	0
S_5 Gazebo, hungry, fresh	0	0	100	100	0	0
S_6 Out, filled, tired	0	0	60	0	0	100

What is the feature vector that defines different states?

Q learning basics

- Q is a quality or utility or Value Function -> helps assess decisions
- Maximizes sum of
  - Immediate reward
  - Projected future reward
- Recursive in nature
- Q must be updated with experience
- Initial Q : all zeros

Initial Utilities – all zeros. Initial State S_3, Discount factor γ = 0.8

	S_1 Tomb, hungry, tired	S_2 Garden, hungry, fresh	S_3 Lake, filled, tired	S_4 Eatery, filled, fresh	S_5 Gazebo, hungry, fresh	S_6 Out, filled, tired
S_1 Tomb, hungry, tired	0	0	0	0	0	0
S_2 Garden, hungry, fresh	0	0	0	0	0	0
S_3 Lake, filled, tired	0	0	0	0	0	0
S_4 Eatery, filled, fresh	0	0	0	0	0	0
S_5 Gazebo, hungry, fresh	0	0	0	0	0	0
S_6 Out, filled, tired	0	0	0	0	0	0

Learning by interacting – Episode 1

- Let first action randomly be $P(S_2 | S_3)$ -> Go to Garden.
- Immediate reward = 70
- Q -> See next state S_2= Discount factor γ*Max{0,0,0,0,0,0} = 0
- Thus new Q(S ,S) = 70 + 0, due to only instant reward

$(S_t \rightarrow S_{t+1})$	**S_1 Tomb, hungry, tired**	**S_2 Garden, hungry, fresh**	**S_3 Lake, filled, tired**	**S_4 Eatery, filled, fresh**	**S_5 Gazebo, hungry, fresh**	**S_6 Out, filled, tired**
S_1 Tomb, hungry, tired	-10	50	70	100	50	0
S_2 Garden, hungry, fresh	20	40	100	100	20	0
S3 Lake, filled, tired	0	70	20	0	0	50
S_4 Eatery, filled, fresh	100	0	0	0	0	0
S_5 Gazebo, hungry, fresh	0	0	100	100	0	0
S_6 Out, filled, tired	0	0	60	0	0	100

Updated Utility: Q’

	S_1 Tomb, hungry, tired	S_2 Garden, hungry, fresh	S_3 Lake, filled, tired	S_4 Eatery, filled, fresh	S_5 Gazebo, hungry, fresh	S_6 Out, filled, tired
S_1 Tomb, hungry, tired	0	0	0	0	0	0
S_2 Garden, hungry, fresh	0	0	0	0	0	0
S_3 Lake, filled, tired	0	70	0	0	0	0
S_4 Eatery, filled, fresh	0	0	0	0	0	0
S_5 Gazebo, hungry, fresh	0	0	0	0	0	0
S_6 Out, filled, tired	0	0	0	0	0	0

Episode 2: Initial state = S_6 Outside

- Exploration: Randomly move to S_3

Reward of 6th row

S_6 Out, filled, tired	0	0	60	0	0	100

- Reward = 60.

Q of 3rd row

S_3 Lake, filled, tired	0	70	0	0	0	0

- Q= 0.8 * max{70,0} = 56
- Updated $Q(S_6, S_3)$ = 60+56=116
- Repeat episodes to reach convergence

Updated Utility

	S_1 Tomb, hungry, tired	**S_2 Garden, hungry, fresh**	**S_3 Lake, filled, tired**	**S_4 Eatery, filled, fresh**	**S_5 Gazebo, hungry, fresh**	**S_6 Out, filled, tired**
S_1 Tomb, hungry, tired	0	0	0	0	0	0
S_2 Garden, hungry, fresh	0	0	0	0	0	0
S_3 Lake, filled, tired	0	70	0	0	0	0
S_4 Eatery, filled, fresh	0	0	0	0	0	0
S_5 Gazebo, hungry, fresh	0	0	0	0	0	0
S_6 Out, filled, tired	0	0	116	0	0	0

Q learning algorithm

For each (*s*,*a*), initialize $\hat{Q}(s,a) = 0$

Observe Current State

Do Forever:

1. Select action *a* and execute it
2. Receive Immediate Reward r
3. Observe new state s’
4. Update Utility Table for $\hat{Q}(s,a)$ as:

$$\hat{Q}(s,a) = r + \gamma \times Max_{a'} Q(s',a')$$

5. Enter New State

Learning Rate

Q[state, action] =

Q[state, action] +

learning Rate *

(reward + Discount Factor * **max**$_i${Q[new_state, S$_i$]}

— Q[state, action])

Discount Factor ⍰ weights future rewards possible, farther the future event, lesser the reward.

Reinforcement Learning:
Each State is a combination of features
Can features change to produce a new state?

- Can an ML agent change features?

  => Yes, provided it interacts with the environment that produces data

- Can features change every now and then by itself?

  => When many entities participate and interact with environment (AI agents/Humans/Nature)

- What applications need this scenario?

  => Where action is needed.

  => Where environment / features are dynamically changing

  => Where there is no training data or human expertise

Applications of RL

- Autonomous car…….What features does it change?
- Trading …… does time series or regression analysis do this? What features change?
- Making decisions at real time using multimedia data ….. A swarm of robots operating
  - In manufacturing
  - In dangerous and unknown areas
- NLP – Text summarization, machine translation, Question Answering, chat-box, dialogue generation
  - supervised DL models to predict words
  - RL through rewards when to look for more words / where to look for important words
  - Need to define reward I terms of linguistic quality parameters such as cohesiveness, understandability etc.
- Dynamic Healthcare – diagnosis / treatment / drugs manufacture/ medical policies
- Auction bidding

What is Reinforcement Learning?

- Agent interacts with Environment
- RL ⍰ Learning from interaction with an environment
- Environment state ? feature-vector
- Long term goal ? maximize predicted cumulative future rewards
- Agent must be able to partially/fully sense the environment state
- Take actions ⍰ change environment state
- Tradeoff between exploration and exploitation
- Needs lots of training

Deep Reinforcement Learning?

- A deep neural network is used to develop either a policy or a value function (state to action/ Q-value) or learn features in complex scenarios
- Deep neural networks require lots of real/simulated interaction with the environment to learn
- Lots of trials/interactions is possible in simulated environments
- We can easily parallelise the trials/interaction in simulated environments

Model-free versus Model-based

- A model of the environment allows inferences to be made about how the environment will behave
- Example: Given a state and an action to be taken while in that state, the model could ***predict the next state and the next reward***
- Models are used for planning, which means deciding on a course of action by considering possible future situations before they are experienced
- Model-based methods use models and planning. Think of this as modelling the dynamics $p(s' \mid s, a)$
- Model-free methods learn exclusively from trial-and-error (i.e. no modelling of the environment)