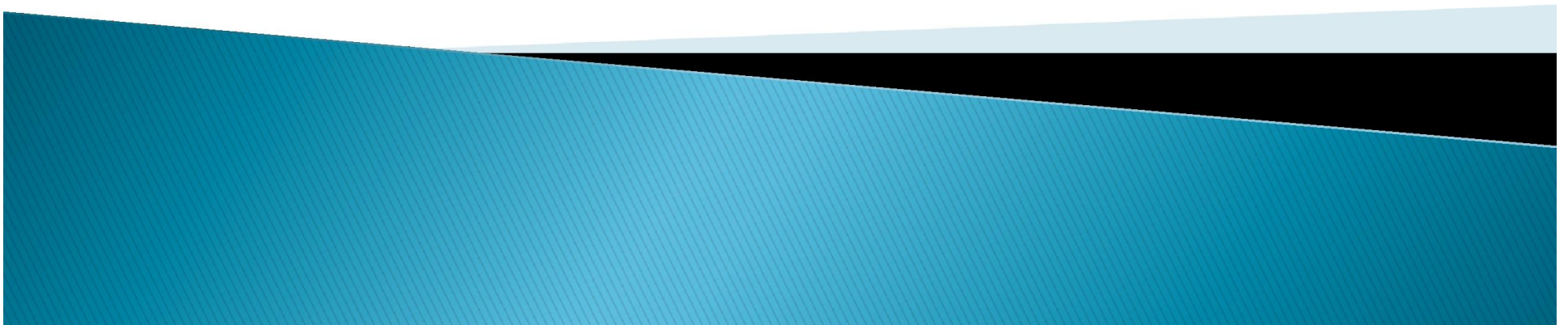


# Reinforcement Learning: Basics

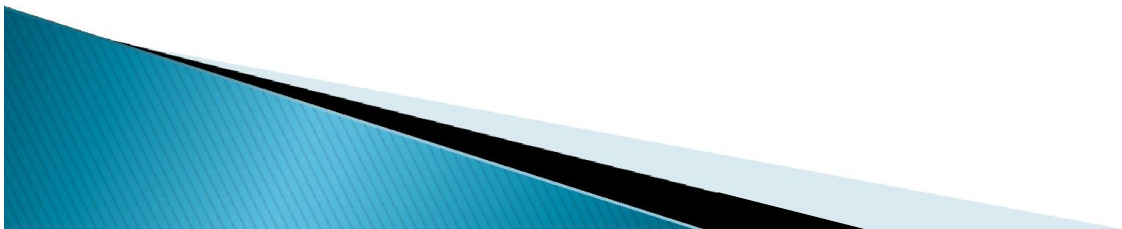
**Nazerfard, Ehsan**  
nazerfard@aut.ac.ir



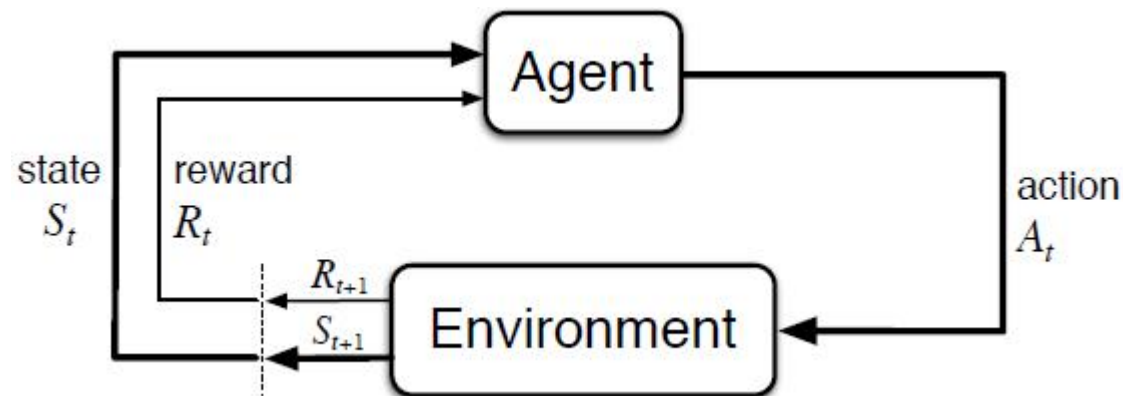
# Learning Types

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Labels provided for every input	No labels provided	Delayed feedback* / Numeric reward signals
Learn from labeled examples	Learn from unlabeled examples	Learn through interactions
One shot decision making	One shot decision making	Sequential decision making

\*Example: Chess game



# The Big Picture



Agent and environment interact at discrete time steps:  $t = 0, 1, 2, \dots$

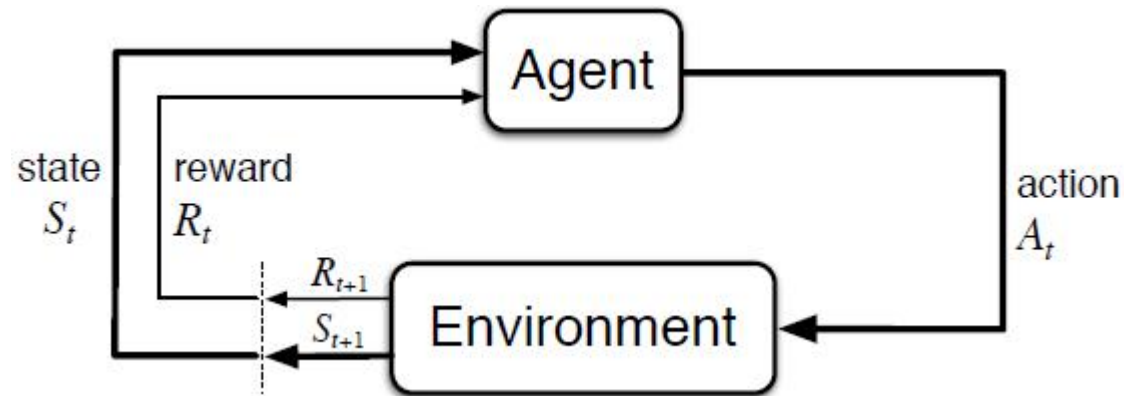
Agent observes state at step  $t$ :  $S_t \in \mathcal{S}$

produces action at step  $t$ :  $A_t \in \mathcal{A}(s)$

gets resulting reward:  $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state:  $S_{t+1}$

# The Big Picture (cont.)



The resulting sequence or trajectory:

$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$

# Agent Learns a Policy

- Policy at step  $t$ ,  $\pi_t$ :
  - a mapping from states to action probabilities
  - $\pi_t(s, a) = \text{probability that } A_t = a, \text{ when } S_t = s$
- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
  - The agent's goal is to get as much reward as it can over the long run.



# Complications

- ❑ The outcome of your actions may be uncertain
- ❑ You may not be able to perfectly sense the state of the world
- ❑ The reward may be stochastic.
- ❑ Reward is delayed (i.e. finding food in a maze)
- ❑ You may have no clue (model) about how the world responds to your actions.
- ❑ You may have no clue (model) of how rewards are being paid off.
- ❑ The world may change while you try to learn it
- ❑ How much time do you need to explore uncharted territory before you exploit what you have learned?

# Returns

- Suppose the sequence of rewards after step  $t$  is:  
 $R_{t+1}, R_{t+2}, R_{t+3}, \dots$

- We want to maximize the expected return,  $E(R_t)$ , for each step  $t$ .

- Discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where  $\gamma, 0 \leq \gamma \leq 1$ , is the discount rate.

# The Markov Property

## □ Markov property

$$p(S_{t+1} = s', R_{t+1} = r \mid a_t, s_t, \mathbf{r_t}, \mathbf{a_{t-1}}, \mathbf{s_{t-1}}, \dots, r_1, a_0, s_0) = \\ p(S_{t+1} = s', R_{t+1} = r \mid s_t, a_t)$$

for all  $s', r$  and histories  $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, a_0, s_0$



# Markov Decision Process

- If a reinforcement learning task has the Markov Property, it is basically a Markov Decision Process (MDP).
- If state and action sets are finite, it is a finite MDP.
- To define a finite MDP, the followings need to be given:

- State and action sets
- Transition probabilities

$$T(s, a, s') = p(S_{t+1} = s' | S_t = s, A_t = a), \forall s, s' \in \mathcal{S}, a \in \mathcal{A}(s)$$

- Reward probabilities

$$R(s, a, s') = \mathbb{E}[S_t = s, A_t = a, S_{t+1} = s'] \\ \forall s, s' \in \mathcal{S}, a \in \mathcal{A}(s)$$

# Value and Q Functions

- The value of a state is the expected return starting from that state; depends on the agent's policy:

$$v^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$$

- The value of taking an action in a state under policy  $\pi$  is the expected return starting from that state, taking that action, and thereafter following  $\pi$ :

$$q^\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

# Bellman Equation for policy $\pi$

□ The basic idea:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

$$v^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma v(s_{t+1}) | S_t = s]$$

$$v^\pi(s) = \sum_a \pi(s, a) \sum_{s'} p(s' | s, a) [R(s, a, s') + \gamma v^\pi(s')]$$

$\pi_t(s, a)$  = probability that  $A_t = a$ , when  $S_t = s$

# Optimality

- There is always at least one (and possibly many) policies that is better than or equal to all the others. This is an optimal policy, denoted by  $\pi^*$ .
- Optimal value function:

$$v^*(s) = \max_{\pi} v^{\pi}(s), \quad \forall s \in \mathcal{S}$$

- Optimal  $q$  function:

$$q^*(s, a) = \max_{\pi} q^{\pi}(s, a), \quad \forall s \in \mathcal{S} \text{ \& } a \in \mathcal{A}(s)$$

# Optimal Value Function

- The value of a state under an optimal policy must equal the expected return for the best action from that state (Bellman equation):

$$v^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} p(s'|s, a) [R(s, a, s') + \gamma v^*(s')]$$

- Similarly for  $q^*$  function:

$$q^*(s, a) = \sum_{s'} p(s'|s, a) [R(s, a, s') + \gamma \max_{a'} q^*(s', a')]$$

# Optimal Policy

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}(s)} \sum_{s'} p(s' | s, a) v^*(s')$$

Or Equivalently:

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}(s)} q^*(s, a)$$

# Recap

$$v^*(s) \leftarrow \max_a \sum_{s'} p(s'|s, a) [R(s, a, s') + \gamma v^*(s')]$$

OR (slightly different notation):

$$v^*(s) \leftarrow \max_a \sum_{s', r} p(s', r|s, a) [r + \gamma v^*(s')]$$

# Value Iteration Algorithm

## □ Idea:

- Start with  $v_0^*(s) = 0$ , which we know is right
- Given  $v_i^*$ , calculate the values for all states for depth  $i+1$ :

$$v_{i+1}(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a) [r + \gamma v_i(s')]$$

(This is called a **value update** or **Bellman update**)

- Repeat until convergence

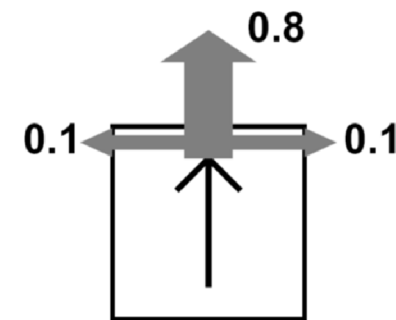
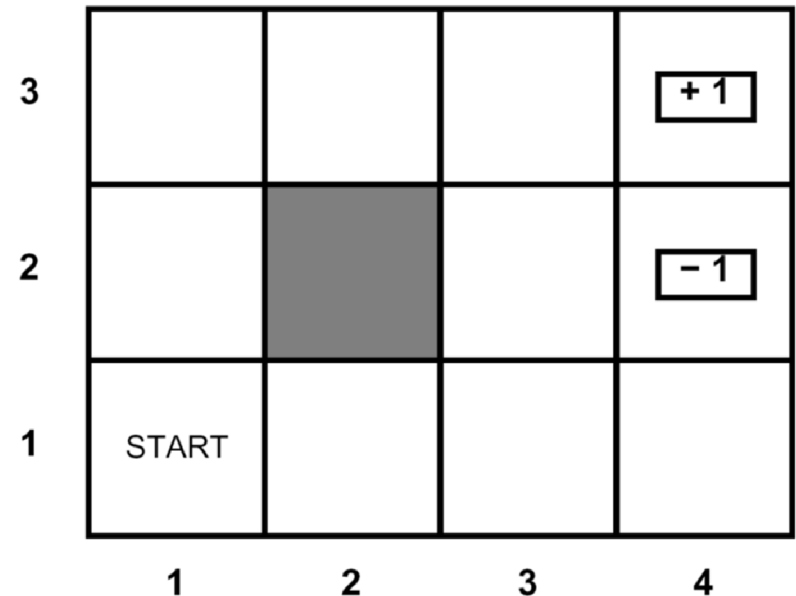
## □ Theorem: It will converge to unique optimal values

- Basic idea: approximations get refined towards optimal values
- Policy may converge long before values do.



# Stochastic Grid World

- ❑ The agent lives in a grid
- ❑ Walls block the agent's path
- ❑ The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North, if there is no wall there.
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays in the same place
- ❑ Small “living” reward each step, e.g.  $R(s) = -0.02$
- ❑ Big rewards come at the end, i.e.  $R(<4,3>) = +1$ ,  $R(<4,2>) = -1$
- ❑ Goal: maximize sum of rewards



# Dynamic Programming for Value Update

- It is assumed that  $r = -0.02$  for all non-terminal states and  $\gamma = 0.9$

$v_0$

3	0	0	0	<span style="border: 1px solid black; padding: 2px;">+1</span>
2	0		0	<span style="border: 1px solid black; padding: 2px;">-1</span>
1	0	0	0	0
	1	2	3	4

$v_1$

3	0	0	0.796	<span style="border: 1px solid black; padding: 2px;">+1</span>
2	0		0	<span style="border: 1px solid black; padding: 2px;">-1</span>
1	0	0	0	0
	1	2	3	4

# Value Update (cont.)

- Information propagates outward from terminal states and eventually all states have correct value estimates.

$v_1$

3	0	0	0.796	<span style="border: 1px solid black; padding: 2px;">+1</span>
2	0		0	<span style="border: 1px solid black; padding: 2px;">-1</span>
1	0	0	0	0
	1	2	3	4

$v_2$

3	0	0.553	0.867	<span style="border: 1px solid black; padding: 2px;">+1</span>
2	0		0.455	<span style="border: 1px solid black; padding: 2px;">-1</span>
1	0	0	0	0
	1	2	3	4

# Optimal Policy

 $v^*$ 

3	0.86	0.90	0.93	<span style="border: 1px solid black; padding: 2px;">+1</span>
2	0.82		0.69	<span style="border: 1px solid black; padding: 2px;">-1</span>
1	0.78	0.75	0.71	0.49
	1	2	3	4

 $\pi^*$ 

3	→	→	→	<span style="border: 1px solid black; padding: 2px;">+1</span>
2	↑		↑	<span style="border: 1px solid black; padding: 2px;">-1</span>
1	↑	←	←	←
	1	2	3	4

# Policy Iteration Algorithm

- Initialize policy  $\pi$  randomly
- Repeat
  - Let  $v \leftarrow v^\pi$  (solve Bellman equations)
  - Let  $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} p(s'|s, a) v(s')$
  -
- This will make  $v \rightarrow v^*$ ,  $\pi \rightarrow \pi^*$



# Further Reading

- ❑ More advanced topics in Reinforcement Learning (RL)
- ❑ Imitation Learning
- ❑ Deep RL



# References

- ❑ R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction, MIT Press, 1998 (1<sup>st</sup> ed.) & 2018 (2<sup>nd</sup> ed.)
- ❑ T. Mitchell, Machine Learning, Chapter 13, McGraw Hill, 1997.

