REINFORCEMENT LEARNING

Machine Learning

IIIy CSE IDP and IDDMP

Refer : Stephen Marsland

Motivation

- **Gap in Learning Paradigms**
  - Supervised learning: trained on correct answers
  - Unsupervised learning: exploits only similarities to cluster
  - Reinforcement learning sits in between, receiving only success/failure feedback
- **Need for Trial-and-Error**
  - Learner must try different strategies without explicit improvement instructions
  - “Trying out” strategies is equivalent to performing a search over state/action space
- **Role of Search**
  - Search underlies the process of testing strategies to maximize a reward
  - Connects back to general search methods

What Is Reinforcement Learning?

Agent–Environment Interaction

1. **Agent**: the learner making decisions
2. **Environment**: the world providing states and rewards

Reward Function

3. Signals “how good” a chosen strategy is
4. Guides the learner to repeat satisfying actions and discard others

Trial-and-Error & the Law of Effect

5. Actions followed by satisfaction become more likely; those followed by discomfort become less likely
6. Originates from Thorndike (1911): strengthening or weakening of action–situation bonds based on outcome

Delayed Credit Assignment

7. Not always the last action that causes an outcome (e.g. child waving arms before falling)
8. Challenges in determining which earlier actions led to success or failure

Reinforcement Learning Framework

Objective

Map states (inputs) to actions so as to maximize a numerical reward

Agent vs. Environment

Agent: selects actions based on current state

Environment: produces states (sensor readings) and rewards

Robot Example

State: sensor readings (noisy, partial view of world)

Actions: motor commands that move the robot

Reward: task performance (e.g. traversing without crashing)

Delayed Rewards

Reward may arrive long after the causative actions (e.g. reaching maze center)

Must consider both immediate and future expected rewards

Policy & Decision Trade-off

Policy: mapping from state → action

Balance **exploitation** (best known action) vs. **exploration** (trying new actions)

Reinforcement Learning Framework (Robot Example)

FIGURE 11.1 A robot perceives the current state of its environment through its sensors, and performs actions by moving its motors. The reinforcement learner (agent) within the robot tries to predict the next state and reward.

Reinforcement Learning Cycle

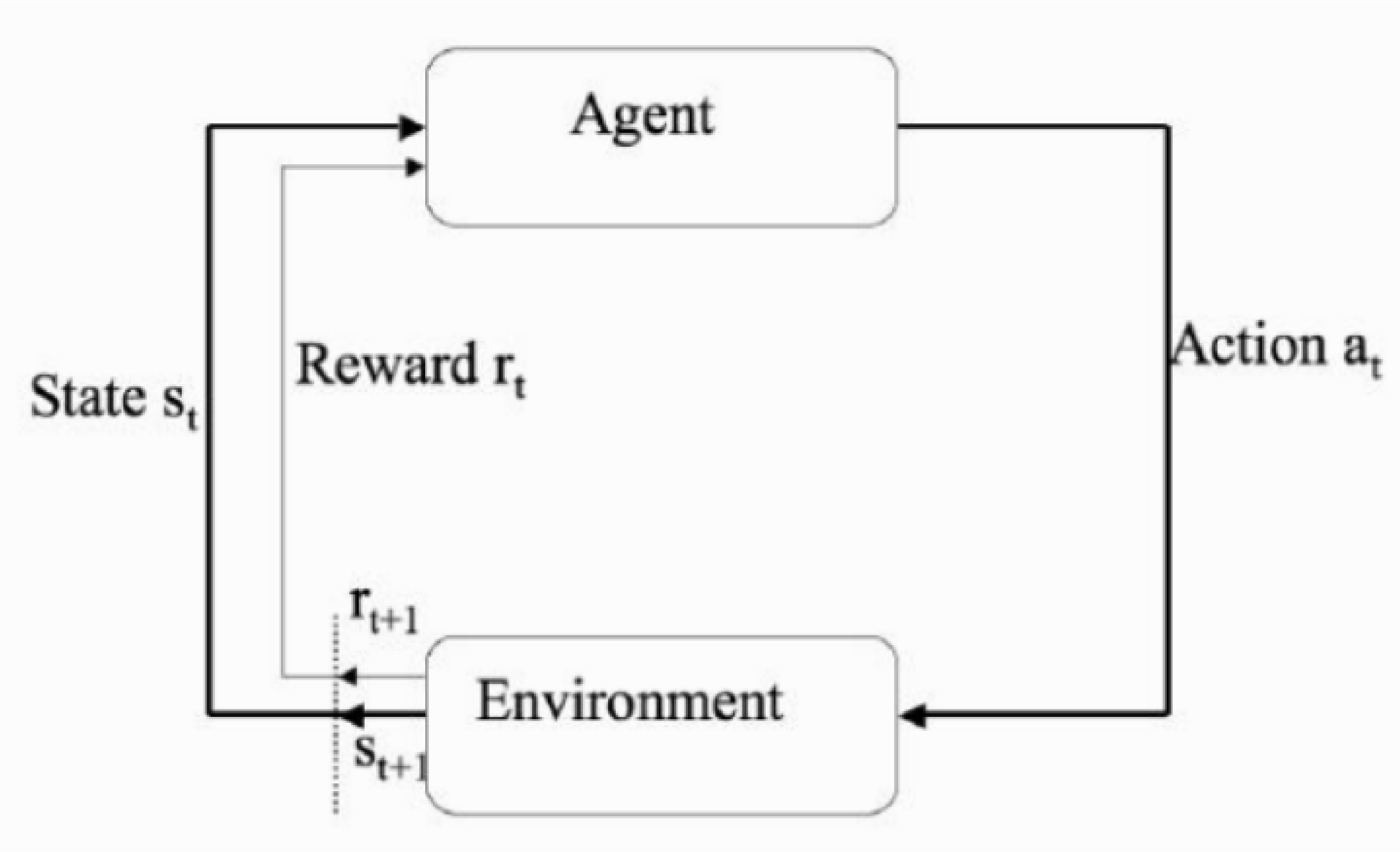

FIGURE 11.2 The reinforcement learning cycle: the learning agent performs action a_t in state s_t and receives reward r_{t+1} from the environment, ending up in state s_{t+1}.

Example – Getting Lost

Scenario

1. Arrive in unfamiliar old-town district at 3 a.m., completely lost
2. Draw map of connected squares (states A–F)
3. Goal: find hostel in square F

Reward Design

4. **Delayed reward**: only upon reaching F (eat all chips)
5. **Stay still** (sleep on feet): reward = −5 (punishment to encourage movement)
6. **All other moves**: reward = 0 (neutral)
7. **Invalid moves** (no direct road): no action possible

Absorbing State

State F is absorbing (once entered, stay and receive final reward)

Old Town Graph

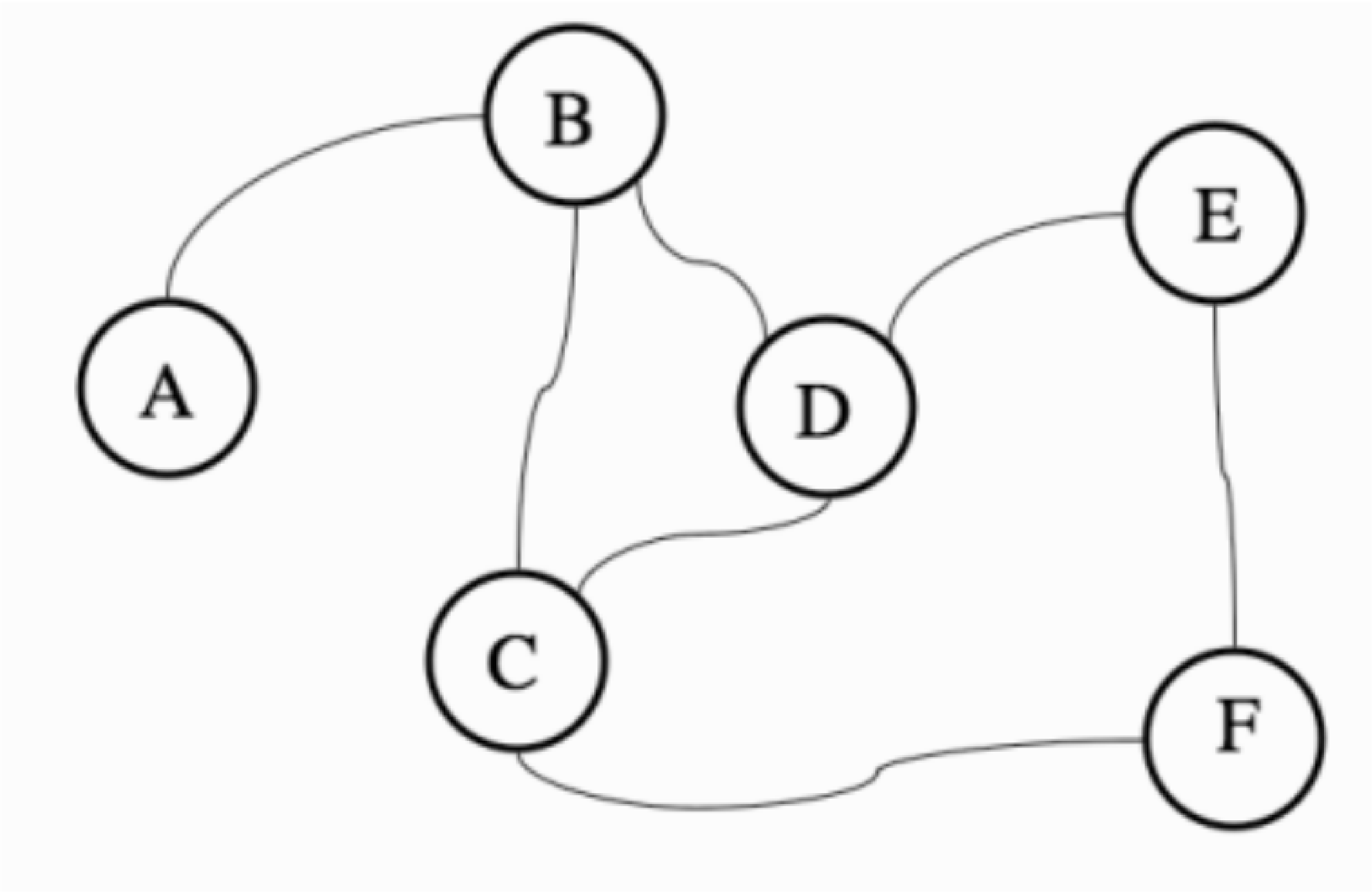

FIGURE 11.3 The old town that you find yourself lost in.

State Diagram of Old Town Graph

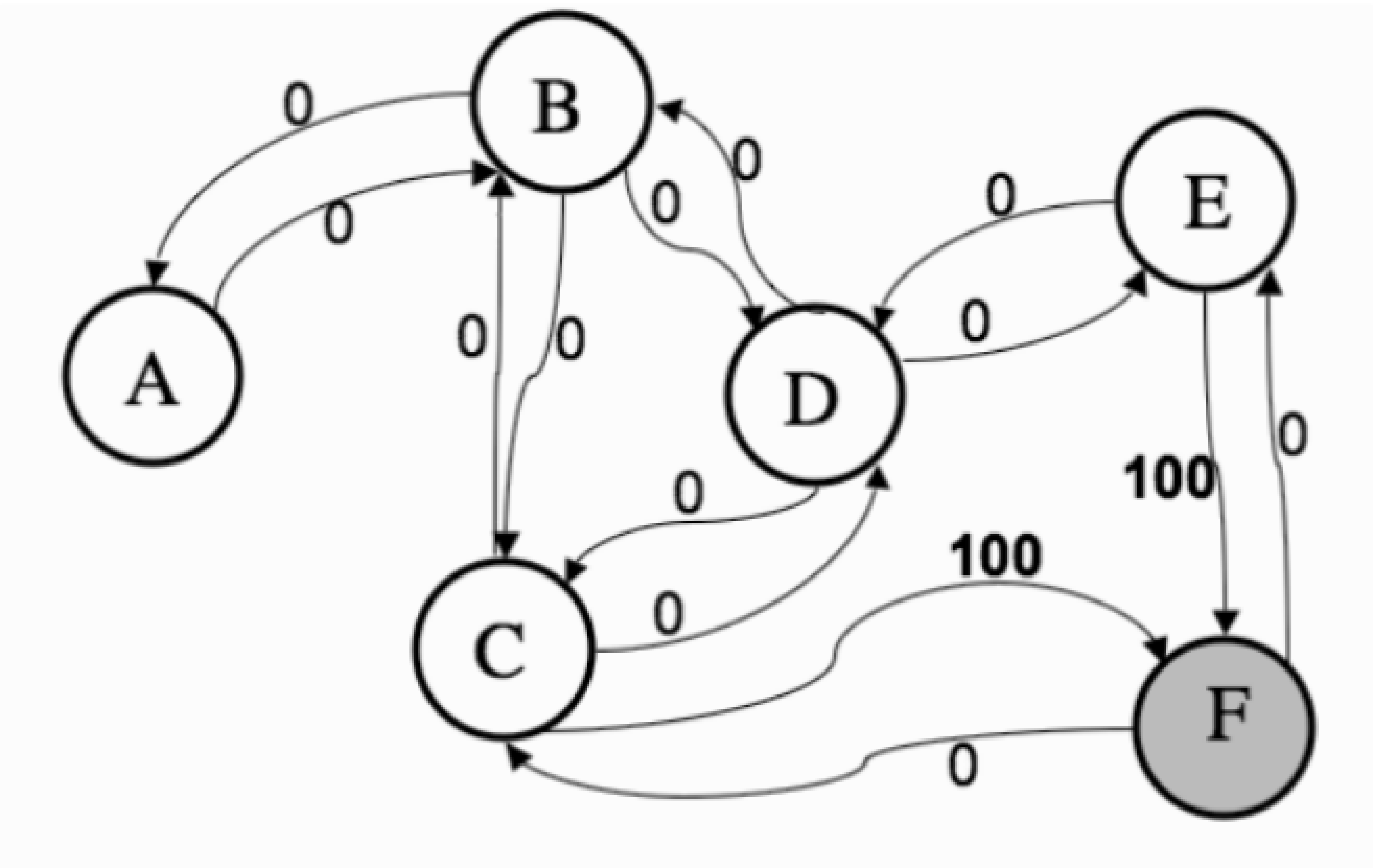

FIGURE 11.4 The *state diagram* if you are correct and the backpacker's is in square (state) F. The connections from each state back into itself (meaning that you don't move) are not shown, to avoid the figure getting too complicated. They are each worth -5 (except for staying in state F, which means that you are in the backpacker's).

Reward Matrix

Current State	Next State					
	A	B	C	D	E	F
A	-5	0	-	-	-	-
B	0	-5	0	0	-	-
C	-	0	-5	0	-	100
D	-	0	0	-5	0	-
E	-	-	-	0	-5	100
F	-	-	0	-	0	-

State & Action Spaces

State Space: all possible states the agent can experience

- Size grows with number of inputs and their ranges
- Example: 5 inputs each 0 – 100 → 100^5 states (huge)

Action Space: all possible actions available in each state

Dimensionality Reduction:

- Quantization can shrink state space
- Example: map each input to two classes (<50, ≥50) → $2^5 = 32$ states

Trade-Off:

- Smaller spaces speed up learning
- Oversimplification may lose important information

Key Insight: carefully choose and balance state/action spaces to make learning tractable without sacrificing accuracy.

Carrots & Sticks – The Reward Function

Purpose: guide the learner by assigning numerical rewards to (state, action) pairs

Design: akin to a fitness function—crafted ad hoc to reflect desired behavior

External Signal: rewards come from the environment, not the learner

Positive & Negative:

- Positive reward → reinforcement of action
- Negative reward (“punishment”) → actions to avoid

Avoid Sub-Goals: extra incentives may be exploited without achieving the real objective

Reward Variants (Maze Example):

- +50 at goal only → learn to reach center
- −1 per move +50 at goal → bias toward shorter paths

Task Types:

- **Episodic:** finite episodes, propagate end reward backward

Action Selection

Estimating Action Value (Q)

- $Q_{s,t}(a)$: average reward observed when taking action a in state s over t trials
- Converges to the true expected reward with enough samples

Greedy

- Always pick the action with the highest $Q_{s,t}(a)$
- Pure exploitation, no exploration

ε-Greedy

- With probability 1−ε : pick the greedy action
- With probability ε: pick a random action → injects exploration

Soft-Max

- Assigns selection probabilities proportional to $\exp(Q_{s,t}(a)/\tau)$
- Temperature τ controls exploration vs. exploitation
  - Large τ: nearly uniform choice
  - Small τ: focuses on higher-valued actions

$$P(Q_{s,t}(a)) = \frac{\exp(Q_{s,t}(a)/\tau)}{\sum_b \exp(Q_{s,t}(b)/\tau)}.$$

Action Selection

Definition: a policy π is a (possibly deterministic) mapping from each state s to an action a

Purpose: defines the agent's behavior by specifying which action to take in every state

Optimal Policy: one that maximizes expected future (discounted) reward

Learning the Policy: core challenge of reinforcement learning

Key Considerations:

- **History Dependence:** how much past information (st_0, st_1, …) is needed to choose the best action now (Markov property)
- **State Valuation:** how to assign a value to the current state to guide policy improvement

Summary & Conclusion

Reinforcement Learning Essence:

- Learner (agent) interacts with an environment, mapping states to actions to maximize rewards

Core Components:

- **State & Action Spaces:** define what the agent perceives and can do
- **Reward Function:** external signal guiding desired behavior
- **Policy:** strategy mapping states to actions

Trade-Offs:

- **Exploration vs. Exploitation:** balance trying new actions and using known good ones
- **Discounting:** prioritizing near-term rewards when future outcomes are uncertain

Design Challenges:

- Crafting appropriate reward functions without unintended shortcuts
- Managing large state/action spaces through abstraction or quantization

Goal:

- Learn an optimal policy that yields the highest cumulative reward over time, adapting through trial and error