ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Artificial Intelligence

Lecturer 14 – Reinforcement Learning

School of Information and Communication Technology - HUST

Reinforcement Learning (RL)

- RL is ML method that optimize the reward
  - A class of tasks
  - A process of trial-and-error learning
    - Good actions are “rewarded”
    - Bad actions are “punished”

Features of RL

- Learning from numerical rewards
- Interaction with the task; sequences of states, actions and rewards
- Uncertainty and non-deterministic worlds
- Delayed consequences
- The explore/exploit dilemma
- The whole problem of goal-directed learning

Points of view

- From the point of view of agents
  - RL is a process of trial-and-error learning
  - How much reward will I get if I do this action?
- From the point of view of trainers
  - RL is training by rewards and punishments
  - Train computers like we train animals

Applications of RL

- Robot
- Animal training
- Scheduling
- Games
- Control systems
- …

Supervised Learning vs. Reinforcement Learning

- Supervised learning
  - Teacher: Is this an AI course or a Math course?
  - Leaner: Math
  - Teacher: No, AI
  - …
  - Teacher: Is this an AI course or a Math course?
  - Leaner : AI
  - Teacher : Yes

- Reinforcement learning
  - World: You are in state 9. Choose action A or B
  - Leaner: A
  - World: Your reward is 100
  - …
  - World: You are in state 15. Choose action C or D
  - Learner: D
  - World : Your reward is 50

Examples

- Chess
  - Win +1, loose -1
- Elevator dispatching
  - reward based on mean squared time for elevator to arrive (optimization problem)
- Channel allocation for cellular phones
  - Lower rewards the more calls are blocked

Policy, Reward and Goal

- Policy
  - defines the agent's behaviour at a given time
  - maps from perceptions to actions
  - can be defined by: look-up table, neural net, search algorithm...
  - may be stochastic
- Reward Function
  - defines the goal(s) in an RL problem
  - maps from states, state-action pairs, or state-action-successor state, triplets to a numerical reward
  - goal of the agent is to maximise the total reward in the long run
  - the policy is altered to achieve this goal

Reward and Return

- The reward function indicates how good things are right now
- But the agent wants to maximize reward in the long-term i.e. over many time steps
- We refer to long-term (multi-step) reward as return

$$R_t = r_{t+1} + r_{t+2} + \ldots + r_T$$

where

- T is the last time step of the world

Discounted Return

- The geometrically discounted model of return

$$R_t = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^T r_T$$
$$0 \le \gamma \le 1$$

- γ is called discount rate, used to
  - Bound the infinite sum
  - Favor earlier rewards, in other words to give preference to shorter paths

Optimal Policies

- An RL agent adapts its policy in order to increase return
- A policy p_1 is at least as good as a policy p_2 if its expected return is at least as great in each possible initial state
- An optimal policy p is at least as good as any other policy

Policy Adaptation Methods

- Value function-based methods
  - Learn a value function for the policy
  - Generate a new policy from the value function
  - Q-learning, Dynamic Programming

Value Functions

- A value function maps each state to an estimate of return under a policy
- An action-value function maps from state-action pairs to estimates of return
- Learning a value function is referred to as the "prediction" problem or 'policy evaluation' in the Dynamic Programming literature

Q-learning

- Learns action-values *Q(s,a)* rather than state-values *V(s)*
- Action-values learning

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(T(s,a),a')$$

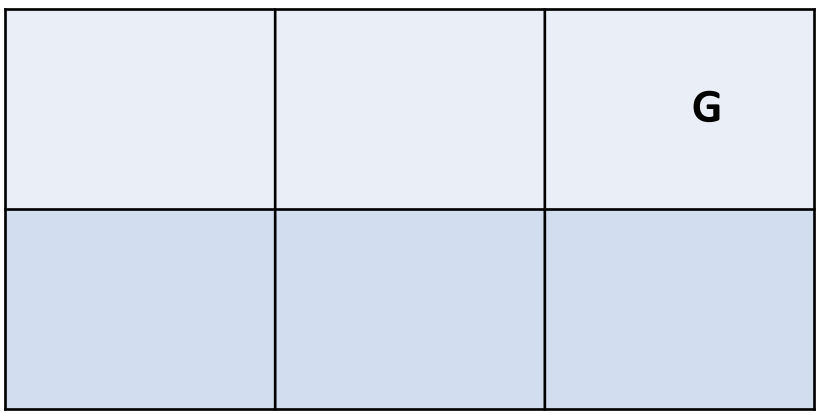
- Q-learning improves action-values iteratively until it converges

Q-learning Algorithm

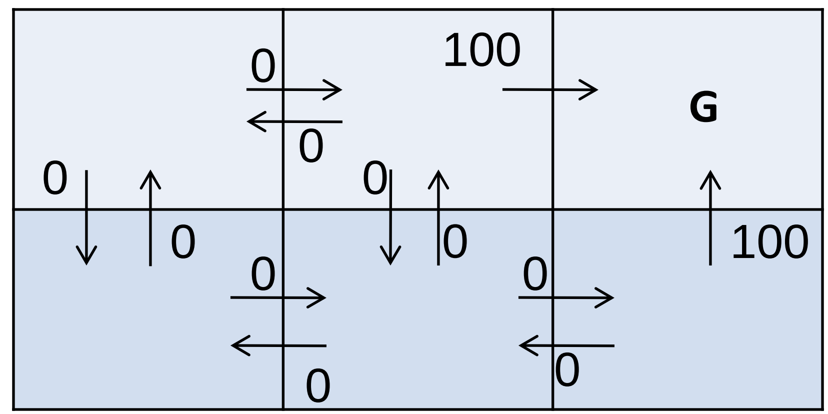
1. Algorithm Q {
2. For each (*s*,*a*) initialize *Q'(s,a)* at zero
3. Choose current action *s*
4. Iterate infinitely{
5. Choose and execute action *a*
6. Get immediate reward *r*
7. Choose new state s'
8. Update *Q'(s,a)* as follows: $Q'(s,a) \leftarrow r + \gamma \max_{\acute{a}} Q'(s',a')$
9. $s \leftarrow s'$
10. }
11. }

Example

- Initially



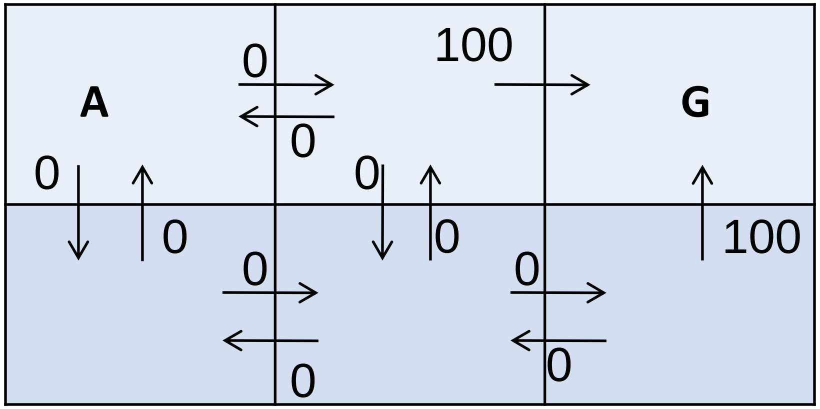
- Initialization

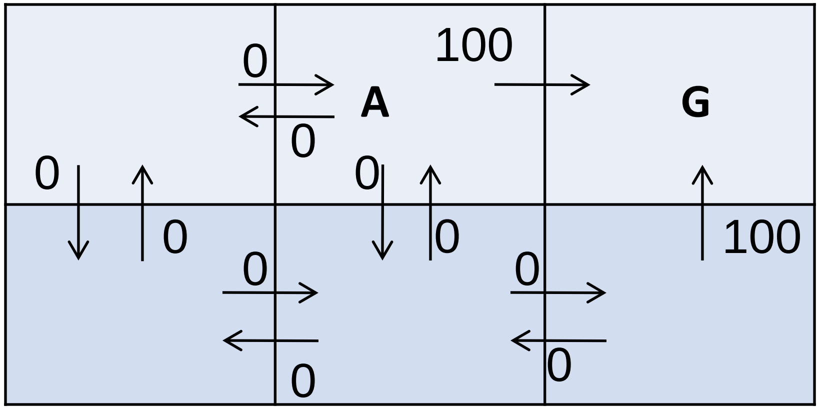


Example

- s_1

- Assume $\gamma = 0,9$
- Go right: s_2
  - Reward: 0

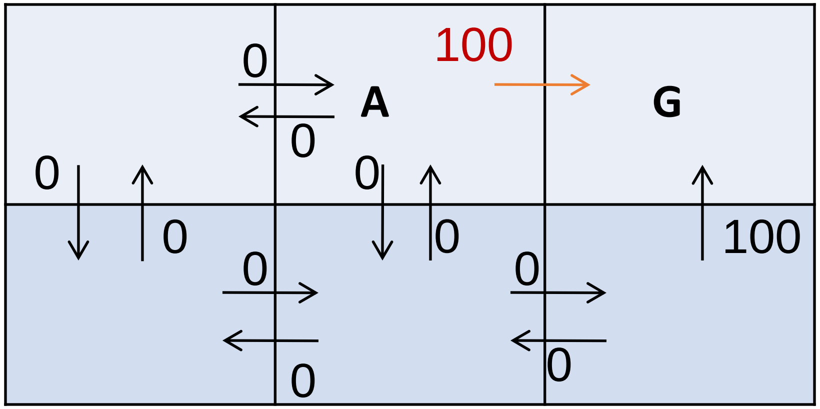


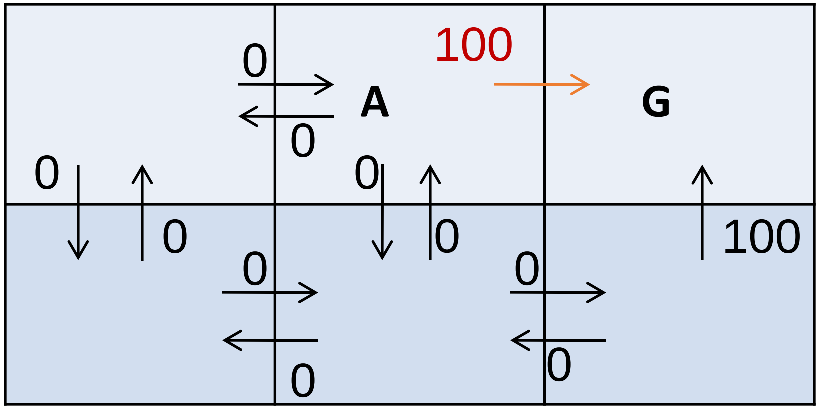


Example

- Go right
  - Reward: 100

- Update s_2
  - Reward: 100

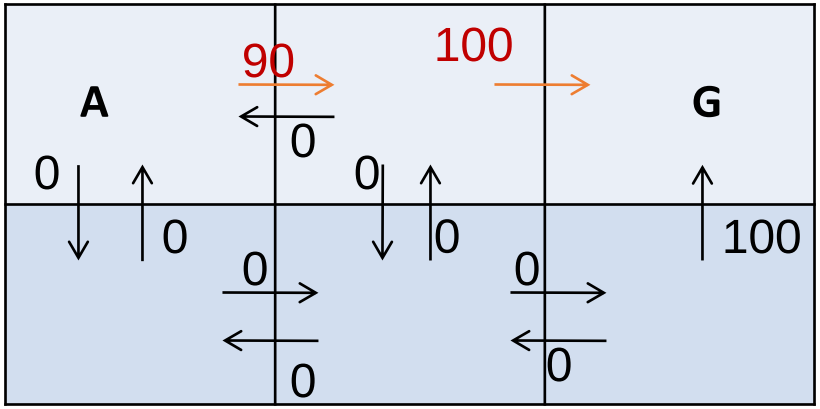


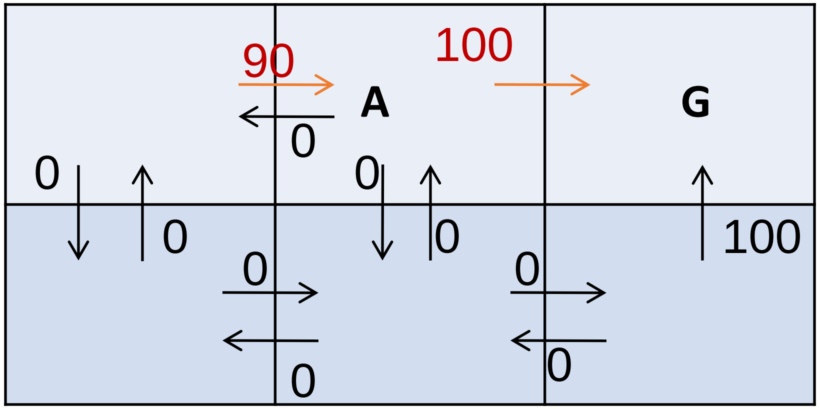


Example

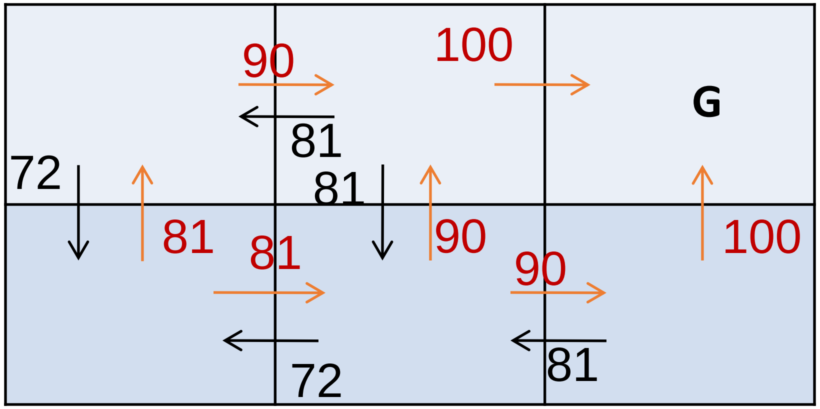
- Update s_1
  - Reward: 90

- s_2



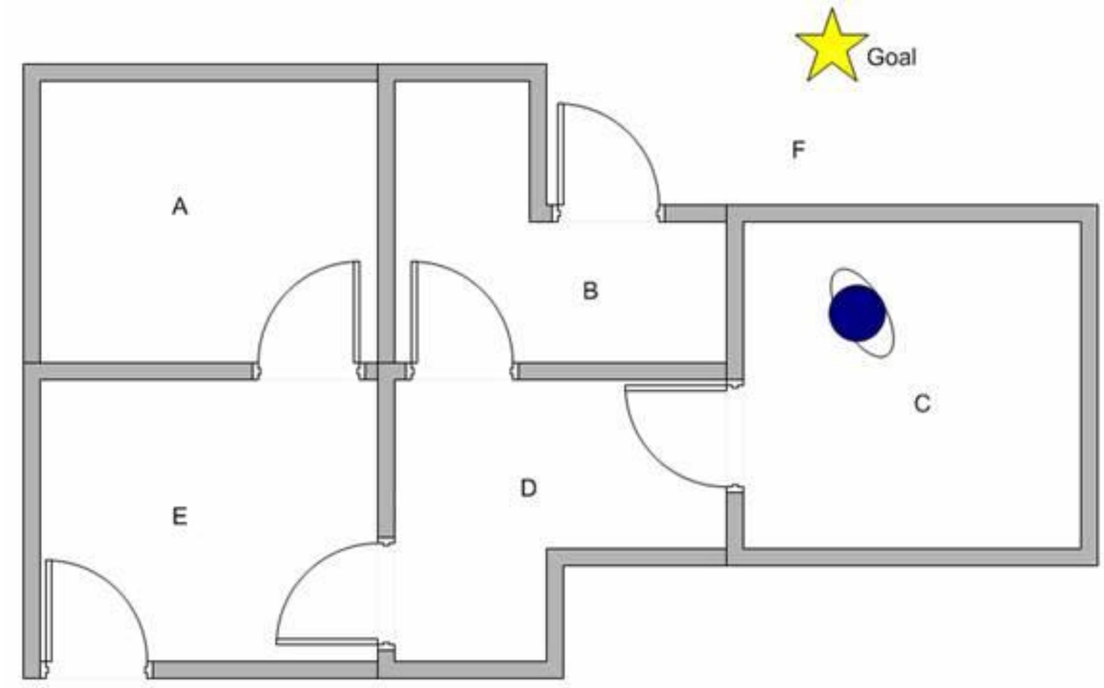


Example: result of Q-learning

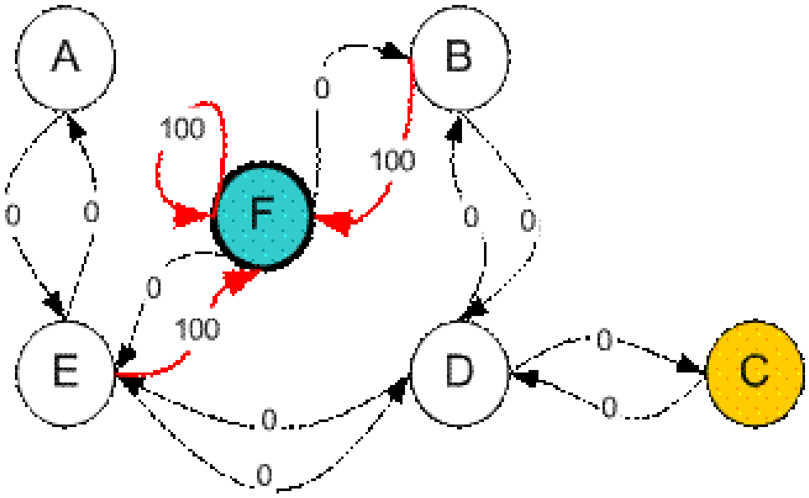


Exercice

- Agent is in room C of the building
- The goal is to get out of the building



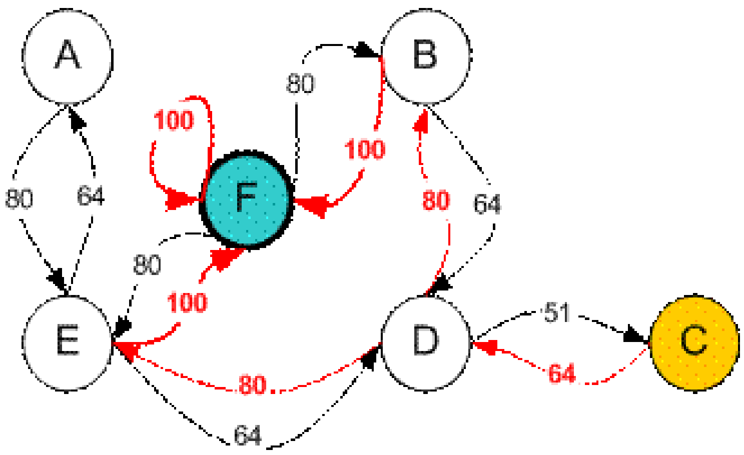
Modeling the problem



	A	B	C	D	E	F
A						
B						100
C						
D						
E						100
F						100

Result

$$\gamma = 0{,}8$$



Divide all rewards by 5

	A	B	C	D	E	F
A					400	
B				320		500
C				320		
D		400	255		400	
E	320			320		500
F		400			400	500

Result: C => D => B => F
C => D => E => F