Reinforcement Learning

Davide Bacciu

Computational Intelligence & Machine Learning Group
Dipartimento di Informatica
Università di Pisa
bacciu@di.unipi.it

Introduzione all'Intelligenza Artificiale - A.A. 2012/2013

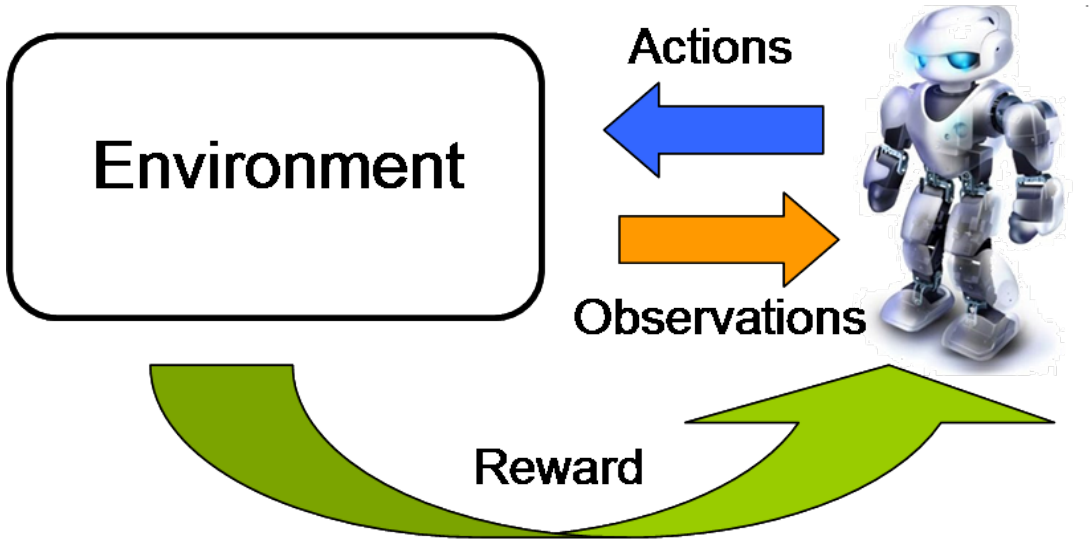
Lecture Outline

Motivation

In some applications it is difficult or impossible to provide the learner with a golden standard (x_n, y_n) (supervised learning)

- Game playing
  - What is the right (target) move given current situation?
  - Easier to provide information about wins and losses
- Optimal control
  - Robot control
  - Planning
- Resource allocation
  - Job scheduling
  - Channel allocation

Reinforcement Learning (RL)



- Learning to chose the best action based on rewards or punishments from the interacting environment
- Actions need to maximize the amount of received rewards

Data Characterization

- Observations s_t
  - The environment state s_t at time t as perceived by the learner
  - Similar to input x_n in supervised/unsupervised learning
  - Assume finite states here
- Actions a_t
  - An action a_t that when performed can alter current state s_t
  - The target action a_t^* is unknown (no supervision)
  - Assume finite actions here
- Rewards $r(s, a)$
  - A reward function $r(s, a)$ that assigns a numerical value to an action a performed in state s
  - Actual rewards cannot be available at any time t

The Reinforcement Learning Task

- A learning agent can perceive a set of environment states S and has a set of actions A it can perform
- At each time instant t the agent is in a state s_t, chooses an action a_t and performs it
- The environment provides a reward $r_t = r(s_t, a_t)$ and produces a new state $s_{t+1} = \delta(s_t, a_t)$
  - r and δ belong to the environment and can be unknown to the agent
- Markov Decision Process (MDP)
  - $r()$ and $\delta()$ depend only on current state s_t and action a_t

Learning Task

Find a policy $\pi : S \rightarrow A$ that will maximize the cumulative reward

What is a Cumulative Reward?

The total reward accumulated over time by following the actions generated by a policy π, starting from an initial state

Several ways of computing cumulated rewards

- Finite horizon reward

$$\sum_{i=0}^{h} r_{t+i}$$

- Average reward

$$\lim_{h\to\infty} \frac{1}{h} \sum_{i=0}^{h-1} r_{t+i}$$

- Infinite discounted reward

$$\sum_{i=0}^{\infty} \gamma^i r_{t+i} \quad s.t. \quad 0 \leq \gamma < 1$$

RL Task Formalization

Assume the infinite discounted reward generated by policy π starting from state s_t

$$V^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i} = \sum_{i=0}^{\infty} \gamma^i r(s_{t+i}, a_{t+i})$$

where γ determines the contribution of delayed rewards Vs immediate rewards

The RL task is to find

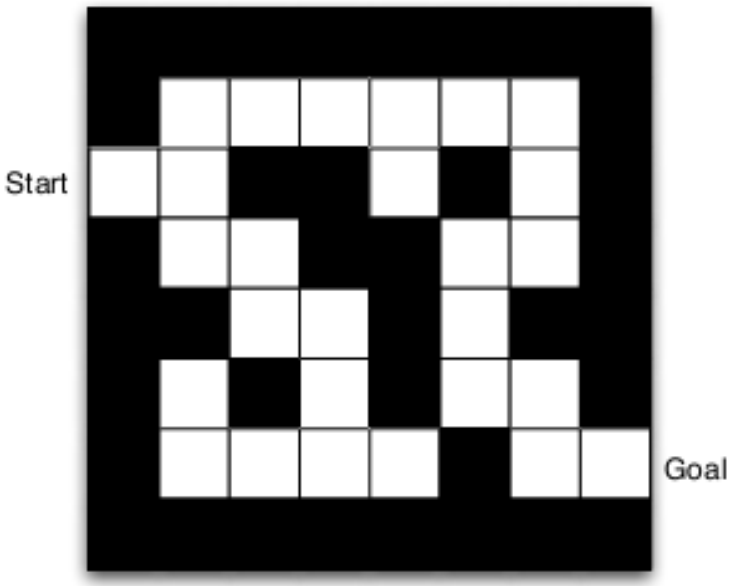
$$\pi^* = \arg\max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

Components of a RL Agent

Any RL agent includes one or several of this components

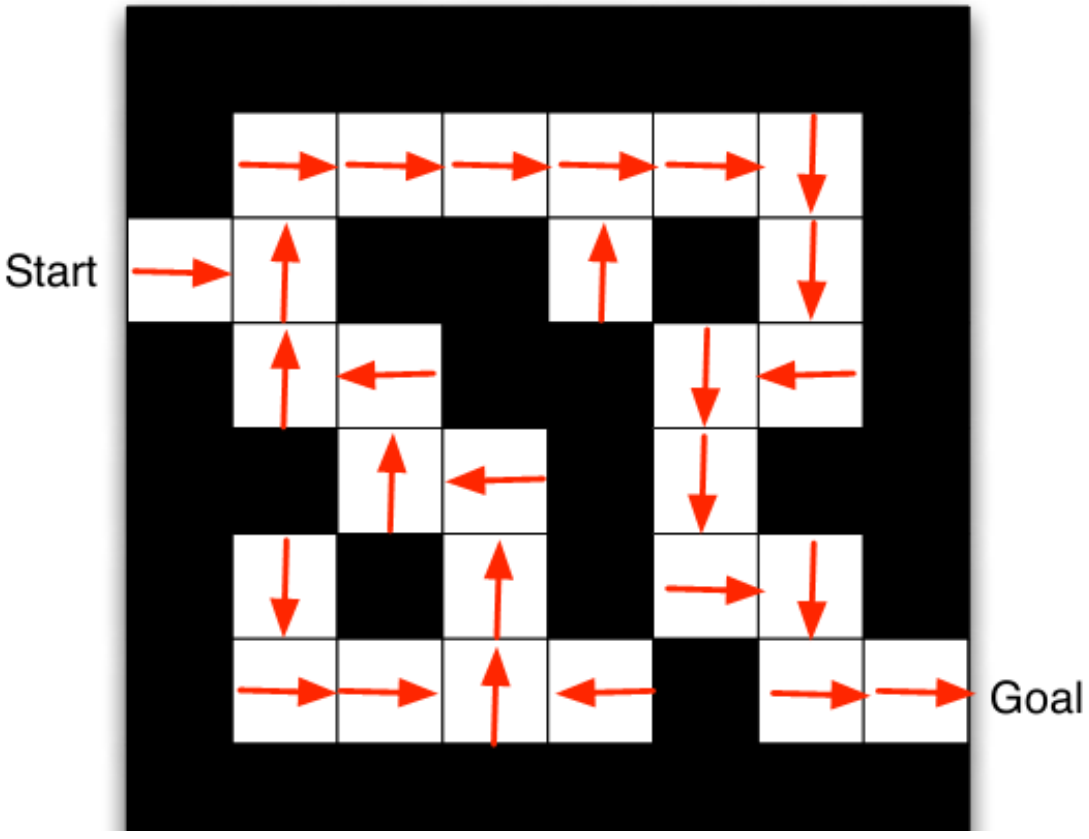
- Policy π
  - Determines the agent behavior
  - What actions are selected in a given state
- Value function $V^{\cdot}(\cdot)$
  - Measures how rewarding is each state and/or action
- Model
  - The agent's internal representation of the environment

Maze Example



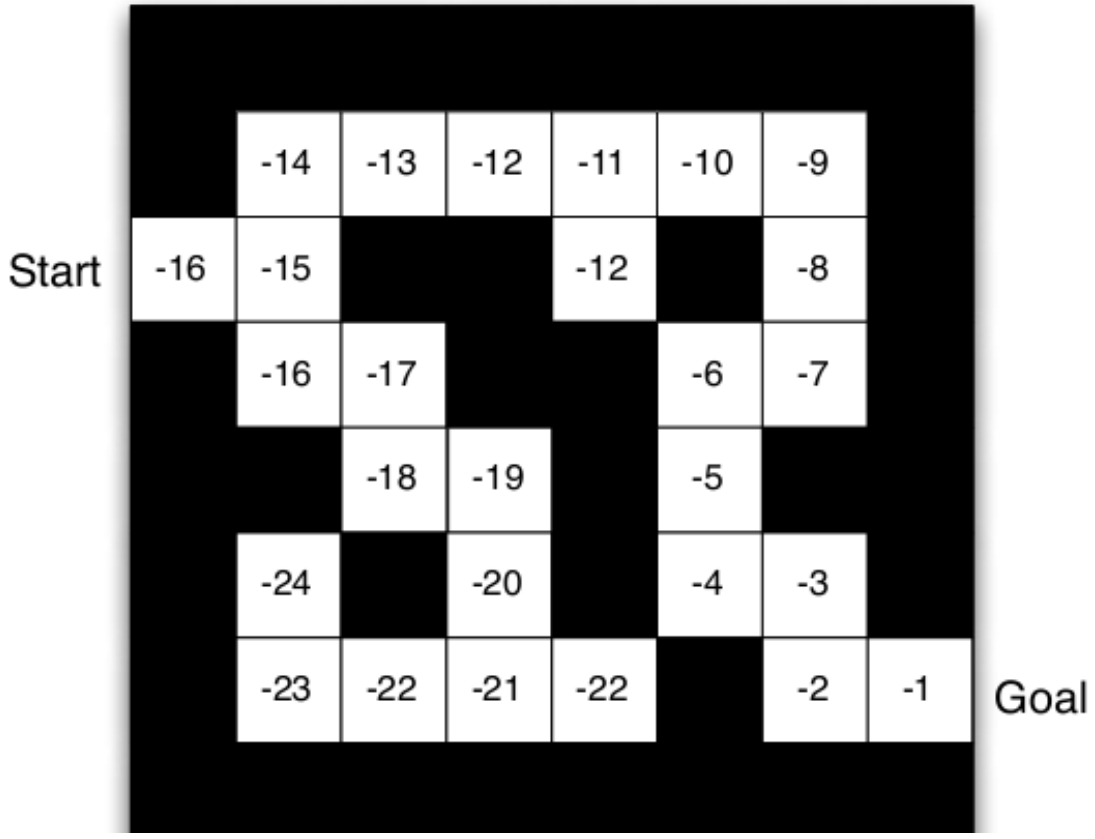
- Environment - a grid with inaccessible points, a starting and a goal location
- States - accessible locations in the grid
- Actions - move N, S, W, E
- Rewards - -1 for each move (time step)

Maze - Policy



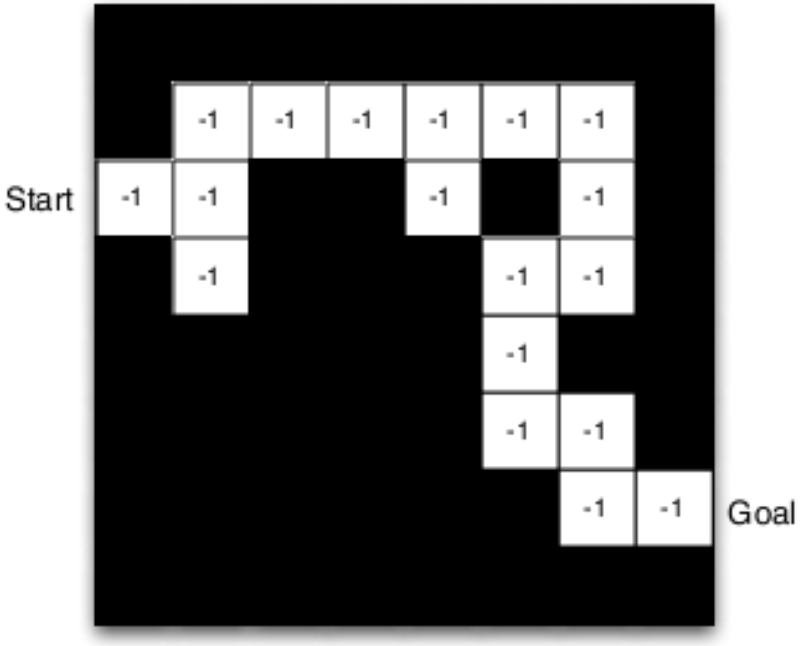
Agent-defined moves in each maze location

Maze - Value Function



A state-dependant value function $V^*(s)$ measuring the time required to reach the goal from each location

Maze - Model



- Agent's internal representation of the environment
  - State change function $s' = \delta(s, a)$
  - Rewards $r(s, a)$ for each state
- Can be based on a very partial and approximated view of the world

Exploration Vs Exploitation

- How does the agent choose the action?
  - Should we always follow only the policy π?
- Exploitation - Choose the action the maximizes the reward according to your learned knowledge
- Exploration - Choose an action that can help in improving your learned knowledge

In general, it is important to explore as well as to exploit

Exploration strategies

How to explore?

- Choose a random action
- Choose an action that has never been selected
- Choose an action that leads to unexplored states

When to explore?

- Until time $T_{explore}$
- ϵ-greedy
- Stop exploring when gathered enough reward

Model-based Vs Model Free

- Model-Based
  - Maximal exploitation of experience
  - Optimal provided enough experience
  - Often computationally intractable
- Model-free
  - Appropriate when model is too complex to store, learn and compute
  - E.g. learn rewards associated to action-states couples (Q function)

The Q-Learning Problem

The RL task is to find the optimal policy π^* s.t.

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

By exploiting the definition of reward and state-dependent value function, we obtain a reformulated problem

$$\pi^* = \arg\max_{a} \left\{ r(s,a) + \gamma V^*(s') \right\} \quad \forall s \in S$$

where $s' = \delta(s,a)$

The Problem

Learn to choose the optimal action in a state s as the action a that maximizes the sum of immediate reward $r(s,a)$ plus the discounted cumulative reward from the successor state

The Optimal Q-function

Define

$$Q^*(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

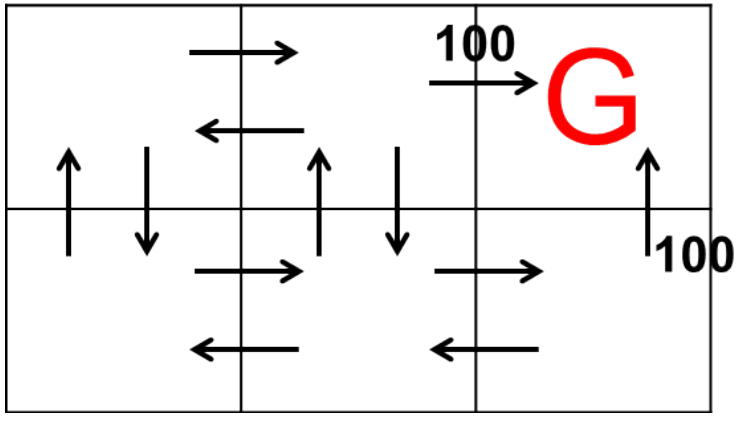
The RL problem for state s becomes

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

$Q^*(s,a)$ denotes the maximum discounted reward that can be achieved starting in state s and choosing action a

Assume that we continue to choose actions optimally

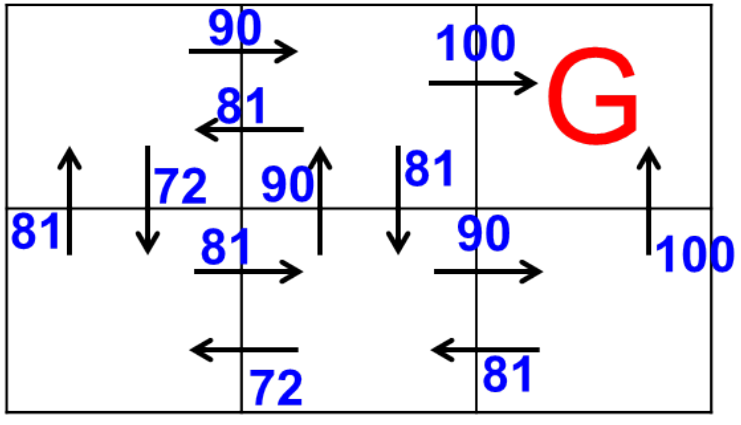
Grid Example ($\gamma = 0.9$)



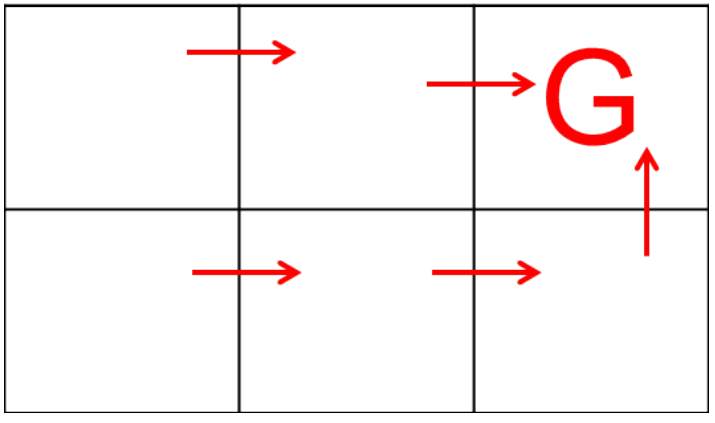
$r(S, A)$

90	100	0
81	90	100

$V^*(s)$



$Q^*(s, a)$



π^*

Q-Learning

What if we don't know the optimal $V^*(s)$?

We don't need to know it, if we assume to know $Q^*(s, a)$, because

$$V^*(s') = \max_{a'} Q^*(s', a')$$

Therefore we can rewrite

$$\begin{aligned} Q^*(s, a) &= r(s, a) + \gamma V^*(\delta(s, a)) \\ &= r(s, a) + \gamma \max_{a'} Q^*(\delta(s, a), a') \end{aligned}$$

A recursive formulation based on Q^* only!

Learning Q

What if we don't known the optimal Q-function?

Key Idea

Use approximated $Q(s, a)$ values that are updated with experience

- Suppose the RL agent has an experience $\langle s, a, r, s' \rangle$
- Experience provides a piece of data ΔQ computed using the current Q estimate

$$\Delta Q = r + \gamma \max_{a'} Q(s', a')$$

- The current Q estimate can be updated using the new piece of data ΔQ

Q Update

A moving average computed each time t a new experience ΔQ arrives

$$Q_{t+1} = (1 - \alpha)Q_t + \alpha \Delta Q_t$$

- The Q terms are
  - $Q_t \rightarrow$ current estimate at time t
  - $Q_{t+1} \rightarrow$ updated estimate
  - $\Delta Q_t \rightarrow$ experience at time t
- $\alpha \in [0, 1]$ is the learning rate
  - Smaller α determine longer term averages
  - Slower convergence but less fluctuations
  - Can be a function of time t

Q-Learning Algorithm

1. Initialize $Q(S, A)$ for all states S and actions A
2. Observe current state s
3. Repeat forever
   - A Select an action a (How?) and execute it
   - B Receive a reward $r(s, a)$
   - C Observe the new state $s' \leftarrow \delta(s, a)$
   - D Update estimate

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left(r(s, a) + \gamma \max_{a'} Q(s', a') \right)$$

   - E Update state $s \leftarrow s'$

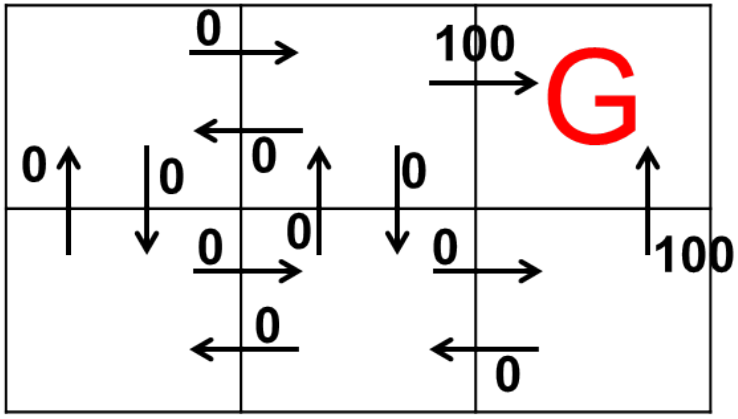
Are Q-Estimates Sufficient?

Theorem (Watkins & Dayan, 1992)

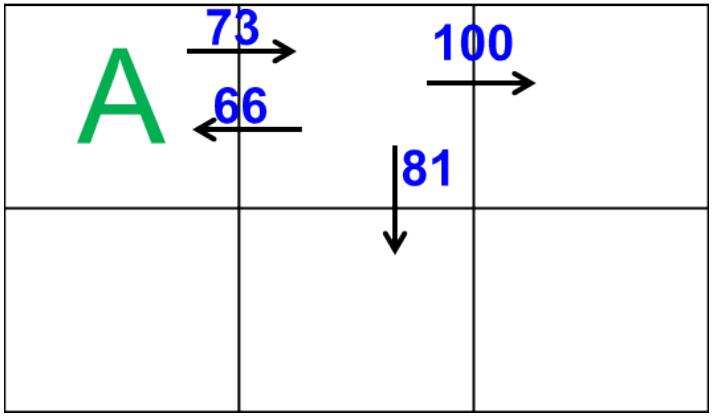
$$Q \longrightarrow Q^*$$

Q-learning will eventually converge to the optimal policy, for any deterministic Markov Decision Process

Grid Example



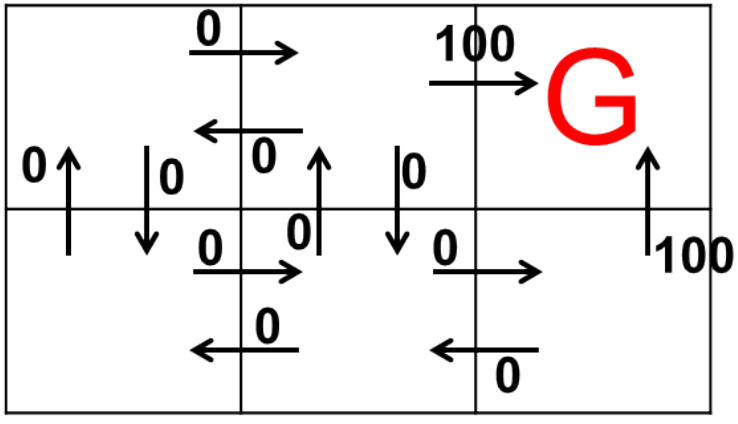
$r(S, A)$



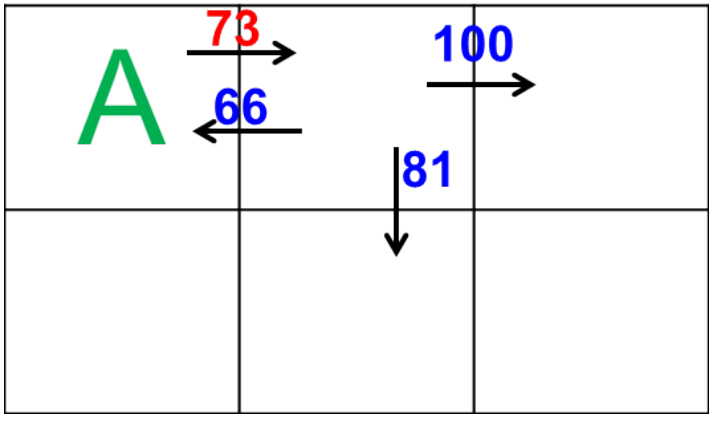
$Q_t(S, A)$

Agent A is in state s_1

Grid Example



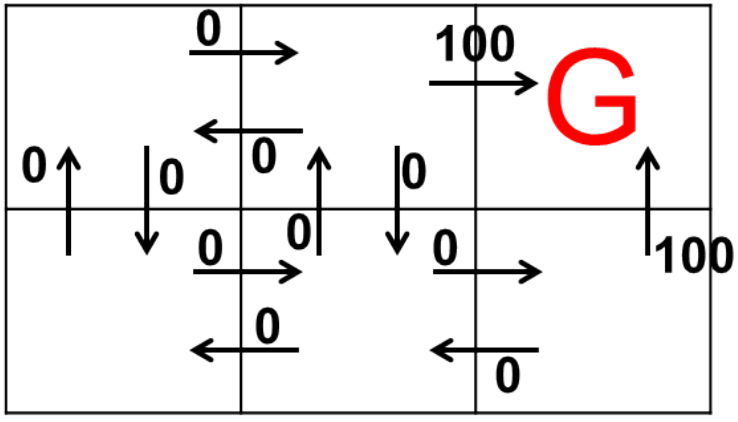
$r(S, A)$



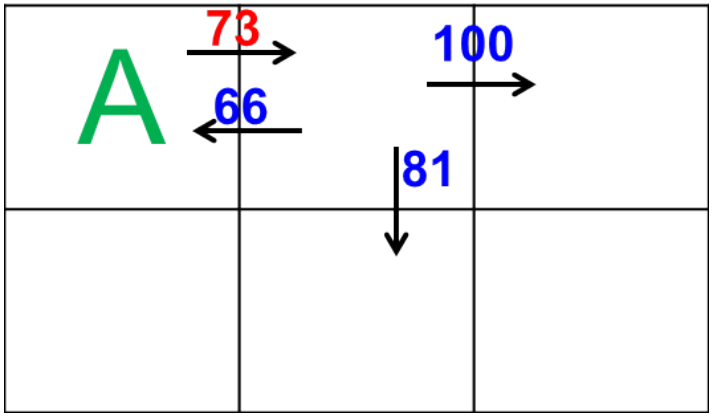
$Q_t(S, A)$

Agent A selects action "move right" a_{right} leading to the new state $s_2 = \delta(s_1, a_{right})$

Grid Example



$r(S, A)$



$Q_t(S, A)$

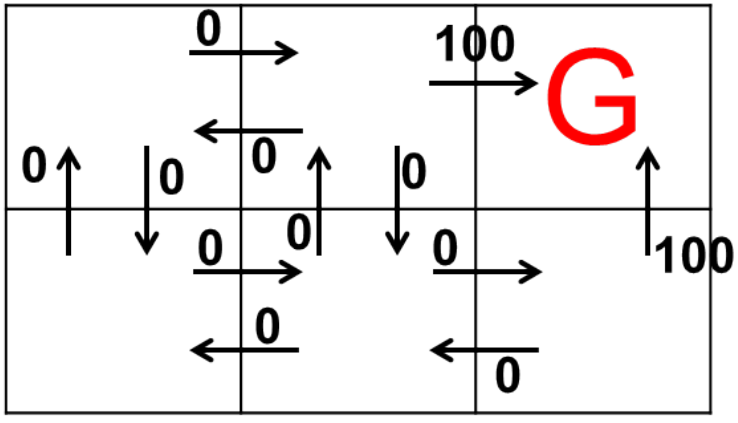
Reward is $r(s_1, a_{right}) = 0$. Assuming $\gamma = 0.9$ and $\alpha = 1$ yields

$$Q_t(s_1, a_{right}) \leftarrow 0 + 1 \cdot r(s_1, a_{right}) + 0.9 \cdot \max_{a'} Q_t(s', a')$$

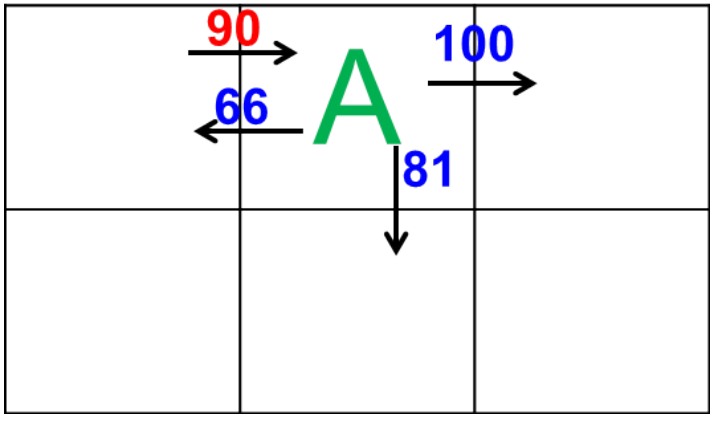
$$\leftarrow 0 + 0.9 \cdot \max_{a'} \{66, 81, 100\}$$

$$\leftarrow 90$$

Grid Example



$r(S, A)$



$Q_{t+1}(S, A)$

The Q-estimate for state s_1 and action a_{right} is updated

Q-Learning Issues (I)

- Delayed feedback
  - Rewards for an action may not be received until several time steps later
  - Credit assignment
- Finite search space
  - Assume states and actions are discrete and finite
- What about generalization?
  - Q-function is basically a lookup-table
  - Unrealistic assumption in large or infinite spaces
  - Generalize from experiences to unseen states

Q-Learning Issues (II)

- Exploration Strategies
  - Convergence to the optimal policy is ensured only if we explore enough
  - Initialize $Q(S, A)$ with values that encourage exploration
  - ϵ-greedy strategy: choose a random action with probability ϵ and the best action with probability $1 - \epsilon$
- Off-policy learning
  - Q-learning learns the value of the optimal policy, no matter what it does
  - Can lead to dangerous policies
  - Looking ahead in time for the next action can be a solution

On-policy Learning

- On-policy procedures learn the value of the policy being followed
- SARSA algorithm does on-policy learning
  - Experience is now $\langle s, a, r, s', a' \rangle$ (SARSA)
  - Looking ahead at the next action a' being performed
  - Next action is chosen on the current policy

SARSA Learning Algorithm

1. Initialize $Q(S, A)$ for all states S and actions A
2. Observe current state s
3. Select action a using a policy based on current $Q(S, A)$
4. Repeat forever
   - A Execute action a
   - B Receive a reward $r(s, a)$
   - C Observe the new state $s' \leftarrow \delta(s, a)$
   - D Select action a' using a policy based on current $Q(S, A)$
   - E Update estimate

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha\left(r(s, a) + \gamma Q(s', a')\right)$$

   - F Update current state $s \leftarrow s'$ and action $a \leftarrow a'$

Take Home Messages

- Reinforcement Learning allows learning when the golden standard is not available
  - An agent interacting with the environment
  - The environment can provide rewards
  - The environment is (partially) observable
- A RL agent includes at least one between
  - Policy
  - Value function
  - Environment model
- Q-Learning
  - Practical and simple algorithm
  - Learning the maximum discounted reward that can be achieved starting in a state s and choosing action a
  - Theoretical guarantees of optimality if the problem is MDP

Take Home Issues

- Exploration Vs Exploitation
  - Choose action to maximize reward
  - Choose action to maximize knowledge acquisition
- More open questions...
  - Credit assignment
  - Continuous state and action spaces
  - Generalization
  - Learning the value function (TD learning)
  - Model-based reinforcement learning