

# Reinforcement Learning

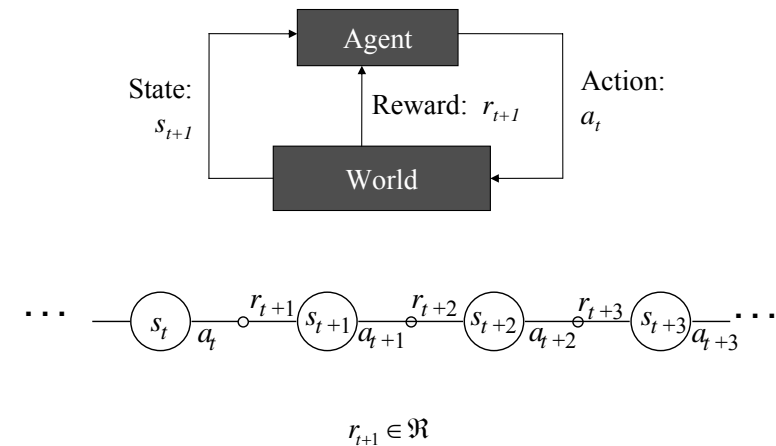
## Goal definition and examples

Mario Martin

Universitat politècnica de Catalunya

Dept. LSI

# The Agent-Environment Interaction



## Goals and Rewards

- Is a scalar reward signal an adequate notion of a goal?—maybe not, but it is surprisingly flexible.
- A goal should specify **what** we want to achieve, not **how** we want to achieve it.
- A goal must be outside the agent's direct control—thus outside the agent.
- The agent must be able to measure success:
  - explicitly;
  - frequently during its lifespan.

## Long Term Reward

- Learning a mapping from situations to actions to maximize long-term reward without using a model of the world
- Long term reward must be defined in terms of the goal of the agent
- Definition of long-term reward must be derived from local rewards

# Finite horizon long-term reward

- First intuitive idea: sum of all rewards obtained.

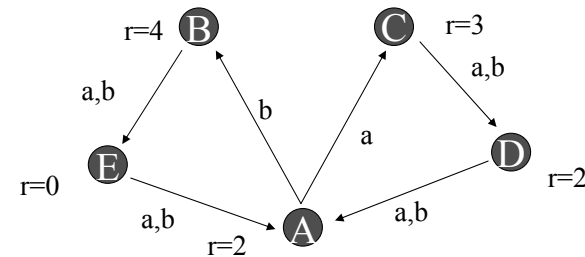
$$\sum_{k=t+1}^{\infty} r_k$$

- Problem: sum must be limited because it can take too large values (infinite values in non-ending tasks)

$$\sum_{k=t+1}^T r_k$$

# Finite Horizon Rewards: Problems

- Limit must be chosen depending on the current task
- Problems: non stationary policies when the limit is not large enough



# Finite Horizon Rewards: Conclusions

- Drawbacks
  - Not useful for non-ending tasks
  - Choose for each task the limit
  - Non stationary when limit is not well chosen
- Nevertheless, finite horizon rewards can be used when:
  - Behavior can be learned in trials
  - Each trial is limited by a max. number of actions to execute and it can be estimated beforehand

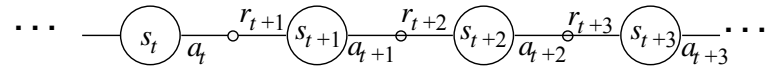
# Discounted Infinite Horizon long-term Reward

- Weighted addition of local rewards

$$\sum_{k=t+1}^{\infty} \gamma^{k-t-1} r_k$$

- The  $\gamma$  parameter ( $0 < \gamma < 1$ ) is called *discounting factor*

# Discounted Infinite Horizon long-term Reward



$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

# Discounted Infinite Horizon Reward

- Advantages

- Finite values for non-ending (or infinite) tasks  
In the best case, (always maximum reinforcement “r”)

$$\sum_{i=1}^{\infty} \gamma^{i-1} r = r \sum_{i=0}^{\infty} \gamma^i = r \frac{1}{1-\gamma}$$

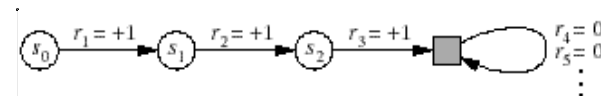
- Greedy policies are always stationary
- Elegant

# Infinite Horizon Rewards

- Problem:
  - Close reinforcements are preferred over delayed reinforcements due to the discounting factor
- This is not always bad. Bias for immediate rewards
- In order to limit this problem, the  $\gamma$  value is chosen very close to 1 (from 0.9 up)

# Infinite Horizon Rewards

- Discounted reward is the usual formalism in reinforcement learning, (also in tasks that can be expressed in trials because they can be reformulated in a continuous infinite trial)
- Discounted reward can cover episodic learning (Think of each episode as ending in an absorbing state that always produces reward of zero)



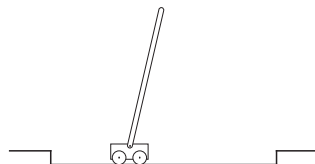
## Differences With Supervised Learning

- No right answer, only evaluation
- No yes/no possibilities (learning values)  
Assume two actions  
 $s, a_1 \rightarrow \text{punish}$  does not imply  $s, a_2 \rightarrow \text{reward}$
- No local evaluation, but global evaluation

## Examples

- Robotics, pole-balancing
- Chess, backgammon, etc
- Industrial applications: (optimization tasks)  
elevator dispatching, job-shop scheduling,  
ATM facturing

### An Example: Pole-balancing

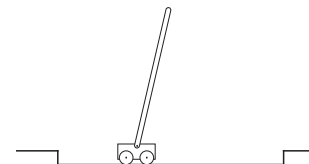


Avoid **failure**: the pole falling beyond a critical angle or the cart hitting end of track.

Situations: Dynamic states of the cart-pole system  
[Discretized velocity, angle, position]

Actions: Push left, Push Righth

### An Example: Pole-balancing



Avoid **failure**: the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:

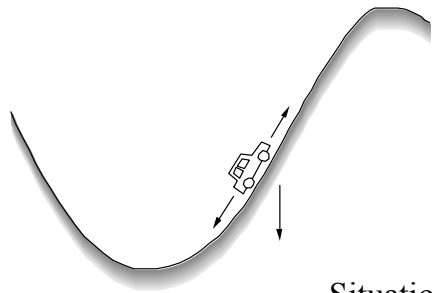
reward = +1 for each step before failure  
 $\Rightarrow$  return = number of steps before failure

As a **continuing task** with discounted return:

reward = -1 upon failure; 0 otherwise  
 $\Rightarrow$  return =  $-\gamma^k$ , for  $k$  steps before failure

In either case, return is maximized by avoiding failure for as long as possible.

## Another example: Mountain Car Problem

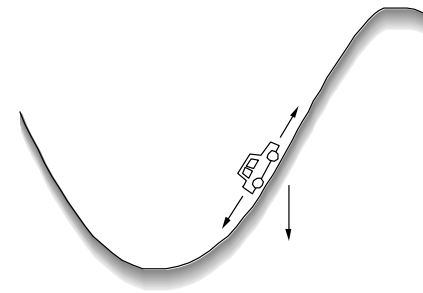


Get to the top of the hill as quickly as possible.

Situations: Car's position and velocity

Actions: forward, reverse, none

## Another example: Mountain Car Problem

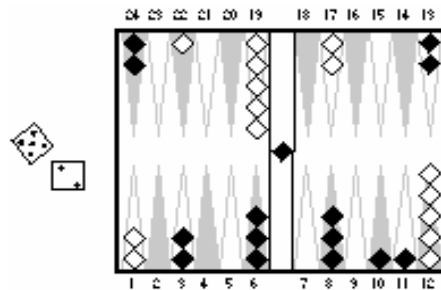


Get to the top of the hill as quickly as possible.

reward = -1 for each step where **not** at top of hill  
 ⇒ return = - number of steps before reaching top of hill

Return is maximized by minimizing number of steps reach the top of the hill.

## Backgammon



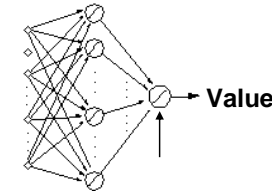
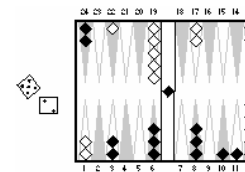
Situations: Configurations of the playing board (about  $10^{20}$ )

Actions: Legal moves at each turn

Rewards: Win +1, Lose -1, else 0

## TD-Gammon

Tesauro, 1992–1995



Action selection by 2–3 ply search

Start with a random network

Play very many games against self

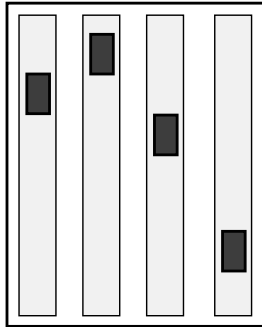
Learn a value function from this simulated experience

**This produces arguably the best player in the world**

# Elevator Dispatching

Crites and Barto, 1996

10 floors, 4 elevator cars



STATES: button states;  
positions, directions, and  
motion states of cars;  
passengers in cars & in halls

ACTIONS: stop at, or go by,  
next floor

REWARDS: roughly, -1 per  
time step for each person  
waiting

Conservatively about  $10^{22}$  states

# Performance Comparison

