Esercitazione 5: Reinforcement Learning

Fondamenti di Intelligenza Artificiale

SAPIENZA
UNIVERSITÀ DI ROMA

A.A. 2022/2023

DIAG, Dipartimento di Ingegneria Informatica, Automatica, e Gestionale "Antonio Ruberti".

Installazione pygame & matplotlib

Installazione con pip:

```
pip3 install pygame matplotlib
```

Installazione con anaconda:

```
conda install pygame matplotlib
```

Si consigliano le versioni python=3.7, pygame=2.3.0, matplotlib=3.5.3

Repository codice esercitazioni

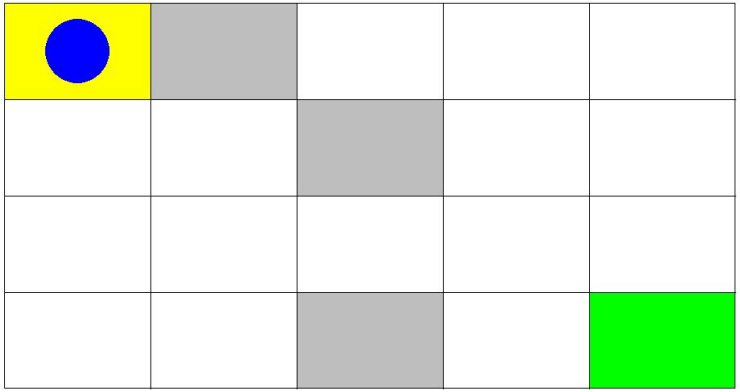
link: https://github.com/KRLGroup/FondamentiIA-2223

Download della repo con git (opzionale):

```
git clone https://github.com/KRLGroup/FondamentiIA-2223.git
```

Il codice per questa esercitazione è in “`esercitazione05.py`”

Environment



- stati: (*x*, *y*) in [0..4] x [0..3] (5*4 = 20 stati)
- 4 azioni: [0, 1, 2, 3]
- reward **deterministico**: 1 se la transizione finisce sulla cella verde (s’ = (4,3)), 0 altrimenti
- transizioni **non deterministiche**: vedi slide successive

Transizioni non deterministiche (1/3)

Azioni **senza rumore** (nota: ordinate in senso orario):

- 0: left
- 1: up
- 2: right
- 3: down

A queste viene aggiunto del rumore (vedi slide successiva) per simulare e.g. attuatori reali di un robot

Transizioni non deterministiche (2/3)

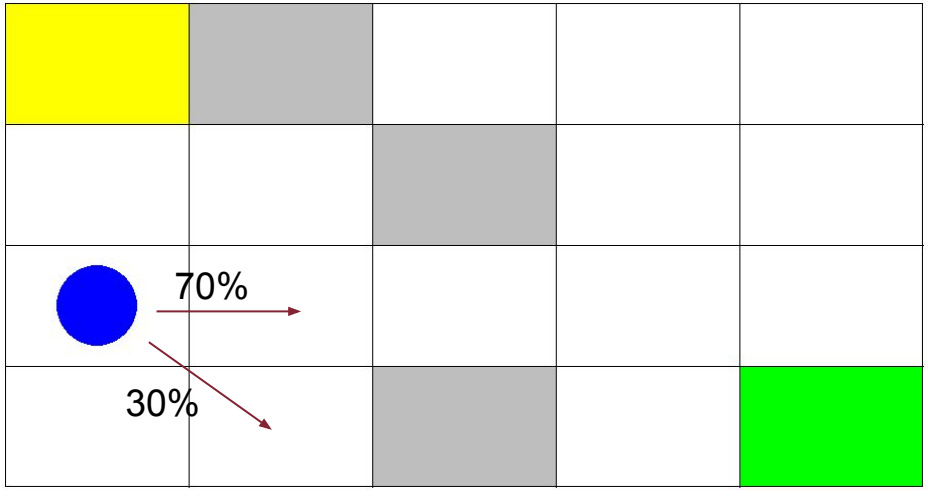
Con probabilità p_{diag}, ogni azione può anche risultare nell'azione diagonale associata data da (nota: ordinate in senso orario):

- left → left + up
- up → up + right
- right → right + down
- down → down + left

Con probabilità 1 - p_{diag} verrà eseguita l'azione non perturbata.

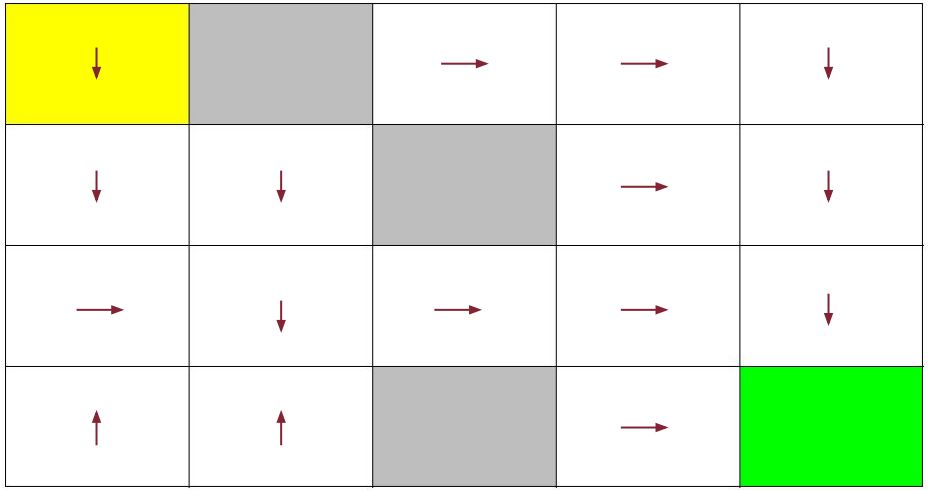
Transizioni non deterministiche (3/3)

Esempio (p_{diag} = 0.3, s = (0,2), a = 2)



Esercizio 1

Dato gamma = 0.99 e la seguente policy π:



Esercizio 1

A) Calcolare $V^{\pi}(s)$ per i seguenti stati:

- $s = (4, 2)$
- $s = (3, 2)$
- $s = (3, 3)$

B) Calcolare $Q^{\pi}(s, a)$ per le seguenti coppie stato-azione:

- $s = (4, 1)$, $a = 1$
- $s = (4, 1)$, $a = 2$

Esercizio 2

A) Implementare Value Iteration (completare codice)

B) Confrontare i valori di $V^{\pi}(s)$ calcolati nell'es. 1.A con quelli $V^{*}(s)$ ottenuti con Value Iteration

Value iteration: algoritmo

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For i = 1, … , H

For all states s in S:

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^*(s')\right]$$

$$\pi_{i+1}^*(s) \leftarrow \arg\max_{a \in A} \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^*(s')\right]$$

This is called a value update or Bellman update/back-up

$V_i^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for i steps

$\pi_i^*(s)$ = optimal action when in state s and getting to act for i steps

$T(s,a,s') = \Pr(s'|s,a)$

Esercizio 3

A) Implementare Q-Learning (completare codice).

Il codice già presente eseguirà 3 run di Q-Learning con 3 diversi valori del learning rate alpha: 0.1, 0.5 e 1.0. Verrà anche mostrato un grafico del reward cumulativo medio (valutato senza esplorazione) dopo ogni episodio di training.

Esercizio 3

B) Quale sembra essere il valore più adatto di alpha per questo problema? Motivare

D) Descrivere e motivare le differenze tra i 3 andamenti del reward cumulativo durante il training, in relazione al valore del learning rate usato.

Q-Learning: algoritmo

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., ε-greedy)
      Take action a, observe r, s'
      Q(s, a) <-- Q(s, a) + α [r + γ max_α,Q(s', a') - Q(s, a)]
      s <-- s';
   until s is terminal
```