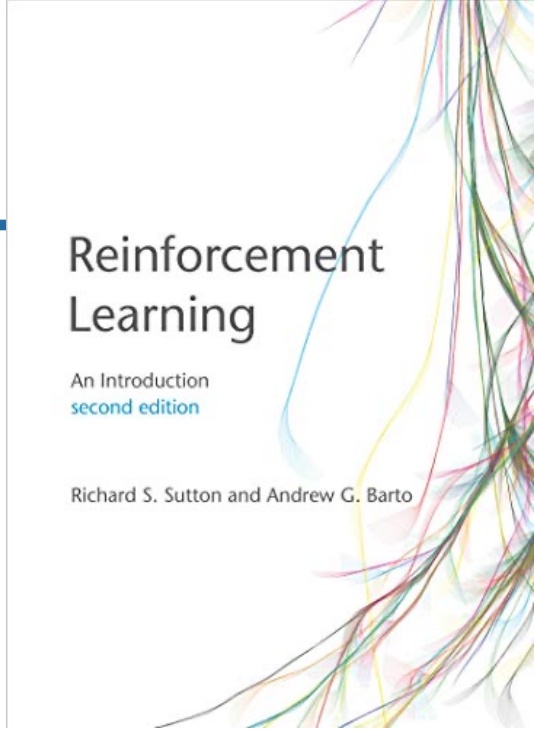
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

# Intelligenza Artificiale

## Reinforcement Learning

a.a. 2022/2023

# Libri di testo



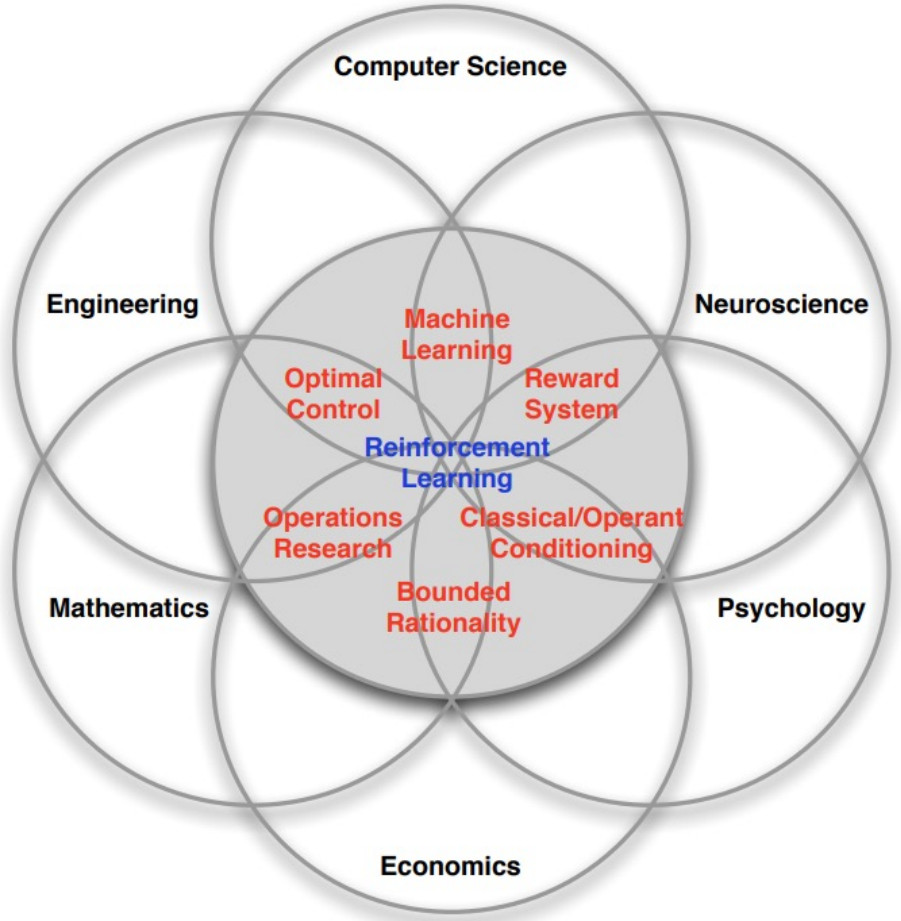

- Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, Second Edition, MIT Press (disponibile online)

- Algorithms for Reinforcement Learning, Szepesvari

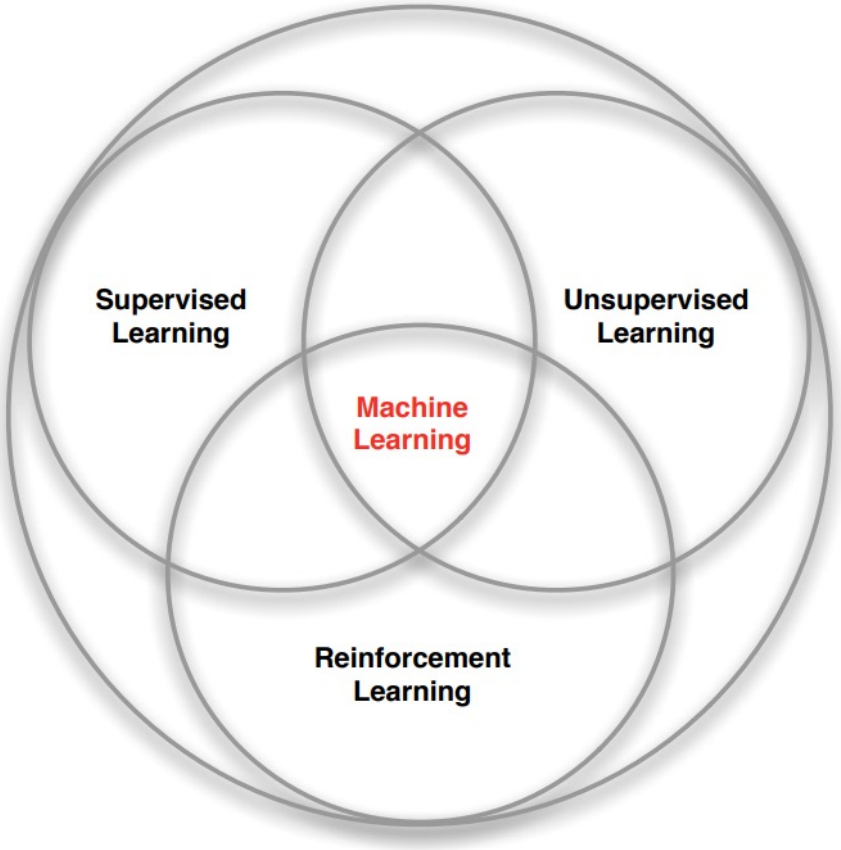
# Outline

- Introduzione al Reinforcement Learning
- Formalizzazione del problema
- Agente basato su RL
- Problematiche

# Posizionare il RL



# Tipi di apprendimento



# Problemi con l'apprendimento supervisionato

- Cosa c'è di sbagliato nell'apprendimento supervisionato?
  - Un agente di apprendimento supervisionato apprende osservando passivamente l'esempio coppie ingresso/uscita fornite da un "insegnante"
  - Possiamo addestrare l'agente di scacchi con l'apprendimento supervisionato?
    - Dati: esempi di posizioni scacchistiche (ognuna etichettata con la corretta mossa)
    - Training data disponibili per i vincitori: $10^8$
    - Lo spazio di tutte le possibili posizioni degli scacchi: $10^{40}$
    - Gli scacchi non possono essere risolti con l'apprendimento supervisionato

# Caratteristiche del RL

- Cosa rende il Reinforcement Learning diverso dagli altri paradigmi di Machine Learning?
  - Non c'è alcuna supervisione, solo *ricompense*
  - Il feedback non è istantaneo
  - Il tempo conta davvero (dati sequenziali, continual learning, non i.i.d)
  - Le azioni dell'agente influenzano le informazioni successive che riceve (non-stazionarietà)

# Esempi di RL

- Imparare a manovrare veicoli
- Impara a controllare robot (camminare, navigare, manipolare)
- Sconfiggere il campione del mondo di Backgammon
- Gestire un portafoglio di investimenti
- Scoprire nuove molecole

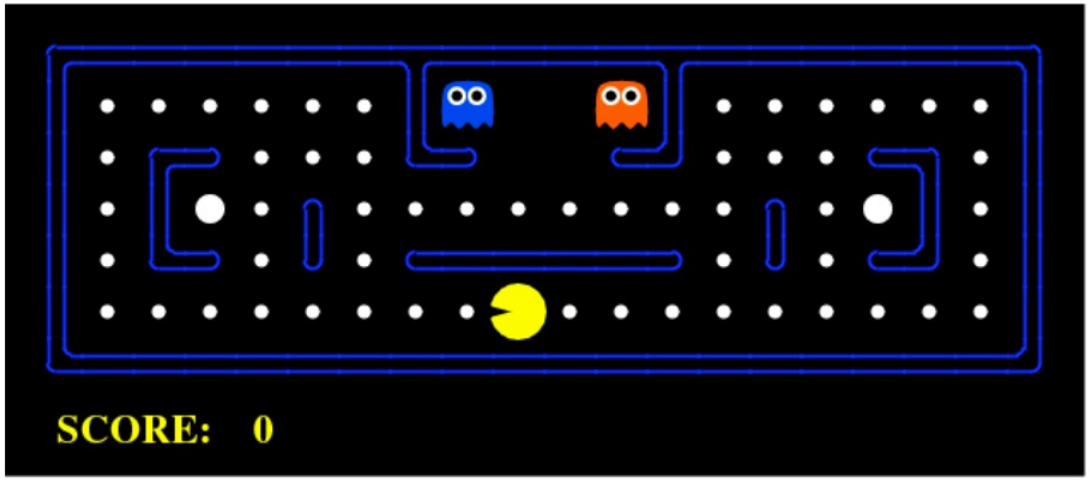
# Cos'è il RL

- Ci sono diversi motivi per apprendere:
  1. Trovare soluzioni
     - Un programma che gioca a scacchi molto bene
     - Un robot industriale con uno specifico scopo
  2. Adattarsi online, gestire situazioni inattese
     - Un programma di scacchi che può adattarsi al giocatore
     - Un robot che può apprendere a navigare in terreni sconosciuti
  - RL può fornire algoritmi per entrambi le situazioni
  - Il secondo punto non è una generalizzazione del primo, è l'apprendere in modo efficiente online, durante le operazioni

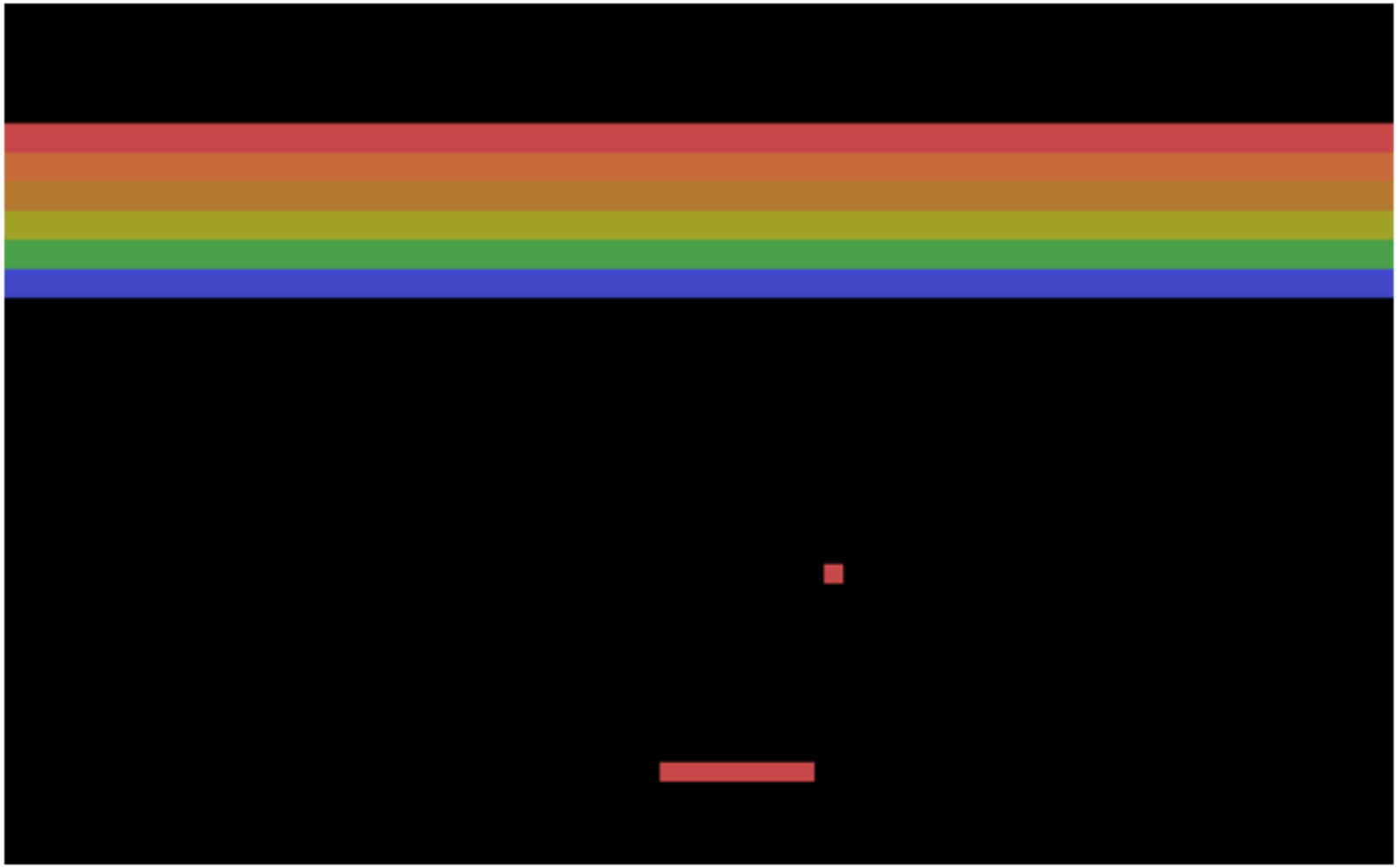
# Cos'è il RL

- Una scienza ed un framework per apprendere le decisioni da prendere dalle interazioni
- Questo richiede di tenere in considerazione
  - …… Il tempo
  - …… le conseguenze delle azioni (a lungo termine)
  - …… acquisire esperienza in modo attivo
  - …… predire il futuro
  - …… gestire le incertezze

# Giocare

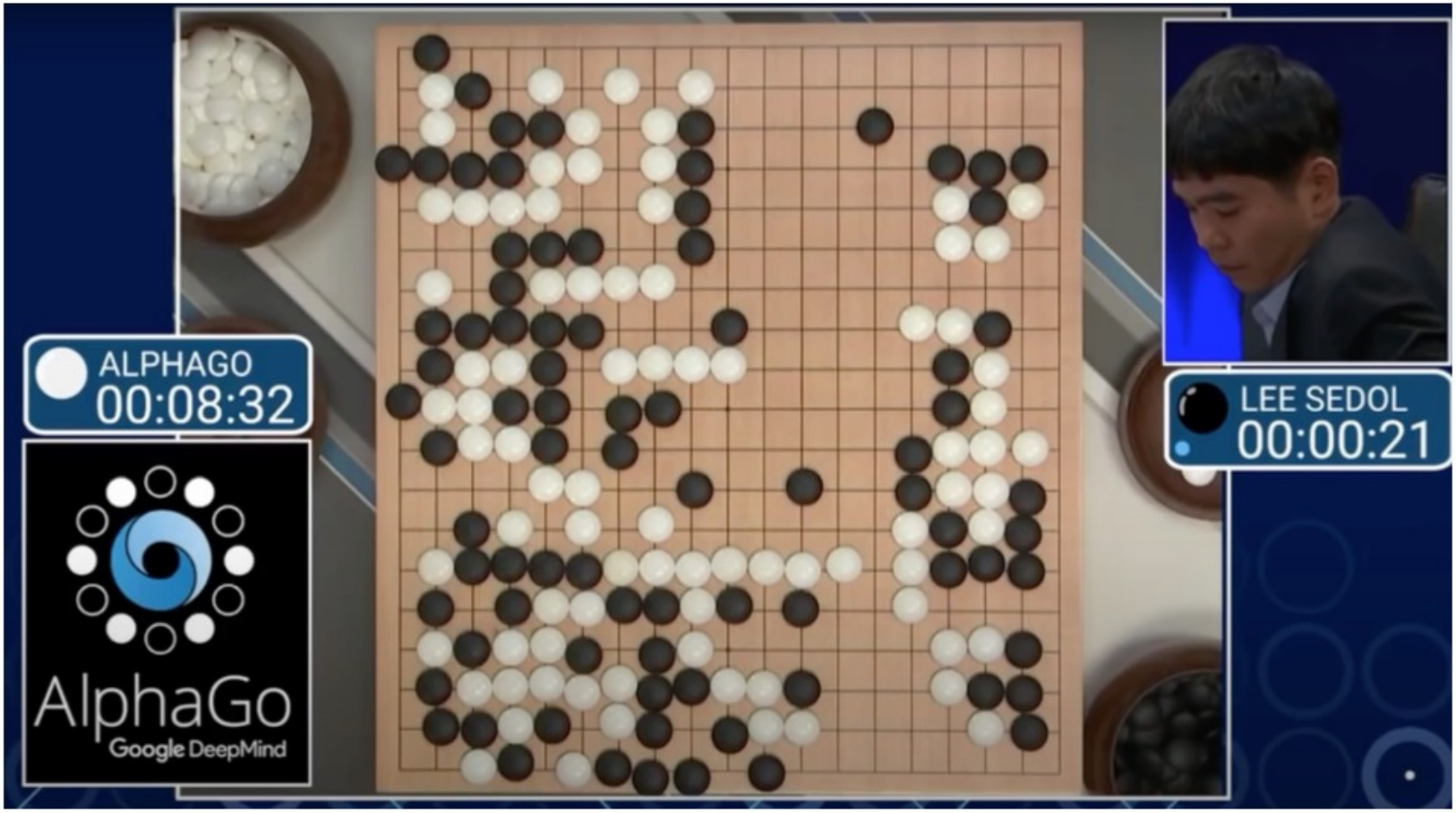



# Giocare



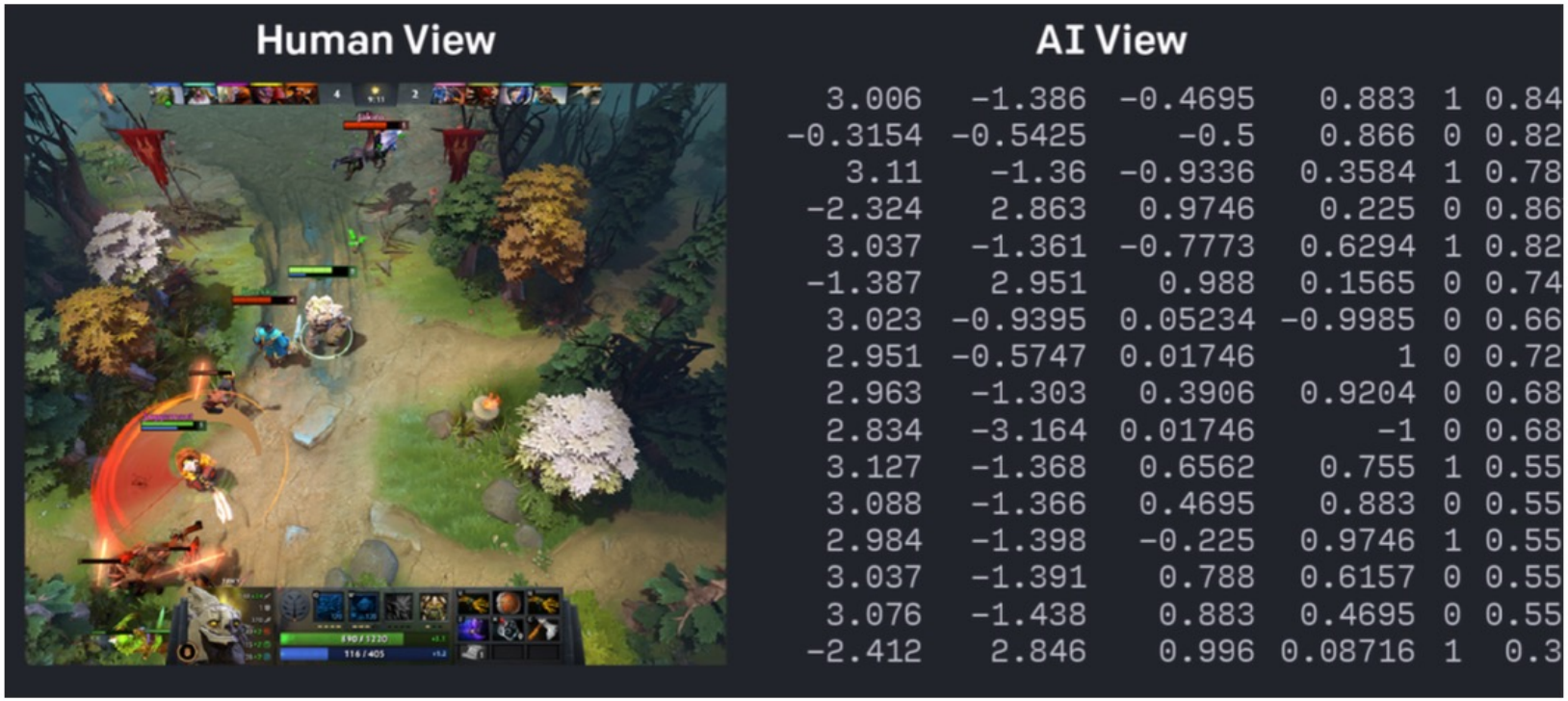
https://www.youtube.com/watch?v=eG1Ed8PTJ18

# Giocare



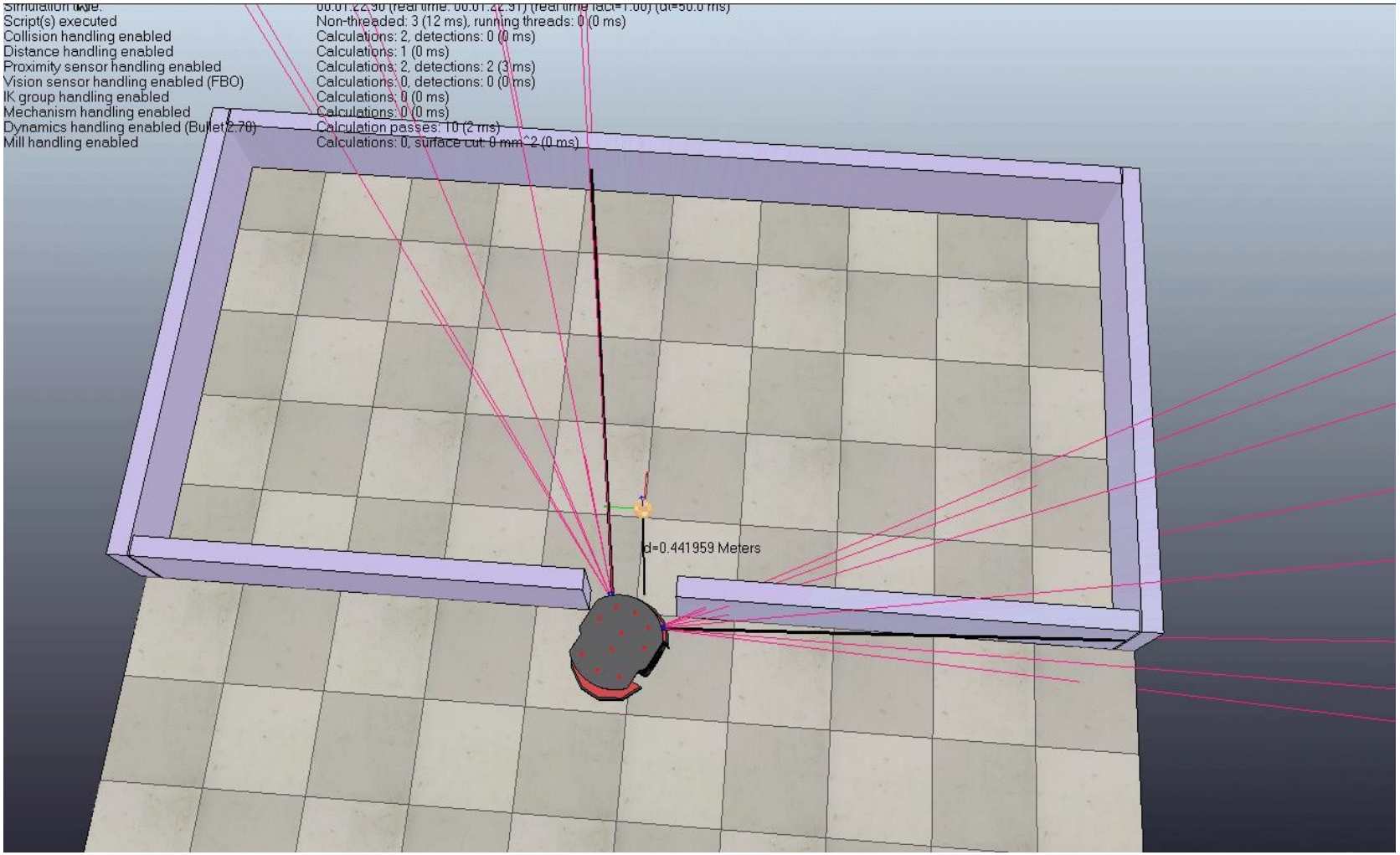

https://www.youtube.com/watch?v=_OV0Hlj8Fb8

# Giocare




https://www.twitch.tv/videos/410533063?t=01h32m02s

# Navigare


Script(s) executed
Collision handling enabled
Distance handling enabled
Proximity sensor handling enabled
Vision sensor handling enabled (FBO)
IK group handling enabled
Mechanism handling enabled
Dynamics handling enabled (Bullet 2.78)
Mill handling enabled
Non-threaded: 3 (12 ms), running threads: 0 (0 ms)
Calculations: 2, detections: 0 (0 ms)
Calculations: 1 (0 ms)
Calculations: 2, detections: 2 (3 ms)
Calculations: 0, detections: 0 (0 ms)
Calculations: 0 (0 ms)
Calculations: 0 (0 ms)
Calculation passes: 10 (2 ms)
Calculations: 0, surface cut: 0 mm^2 (0 ms)
d=0.441959 Meters

# Manipolare

https://www.youtube.com/watch?v=jwSbzNHGflM

# Formalizzare il Reinforcement Learning

# Ricompense (Rewards)

- Una *ricompensa* $R_t$ è un feedback rappresentato con uno scalare
- Indica il grado di efficienza dell'agente allo step *t*
- Il compito dell'agente è quello di massimizzare la ricompensa cumulativa
- Il Reinforcement Learning è basato sulla *reward hypothesis*:

***Tutti** gli obiettivi possono essere descritti come la massimizzazione della ricompensa cumulativa prevista*

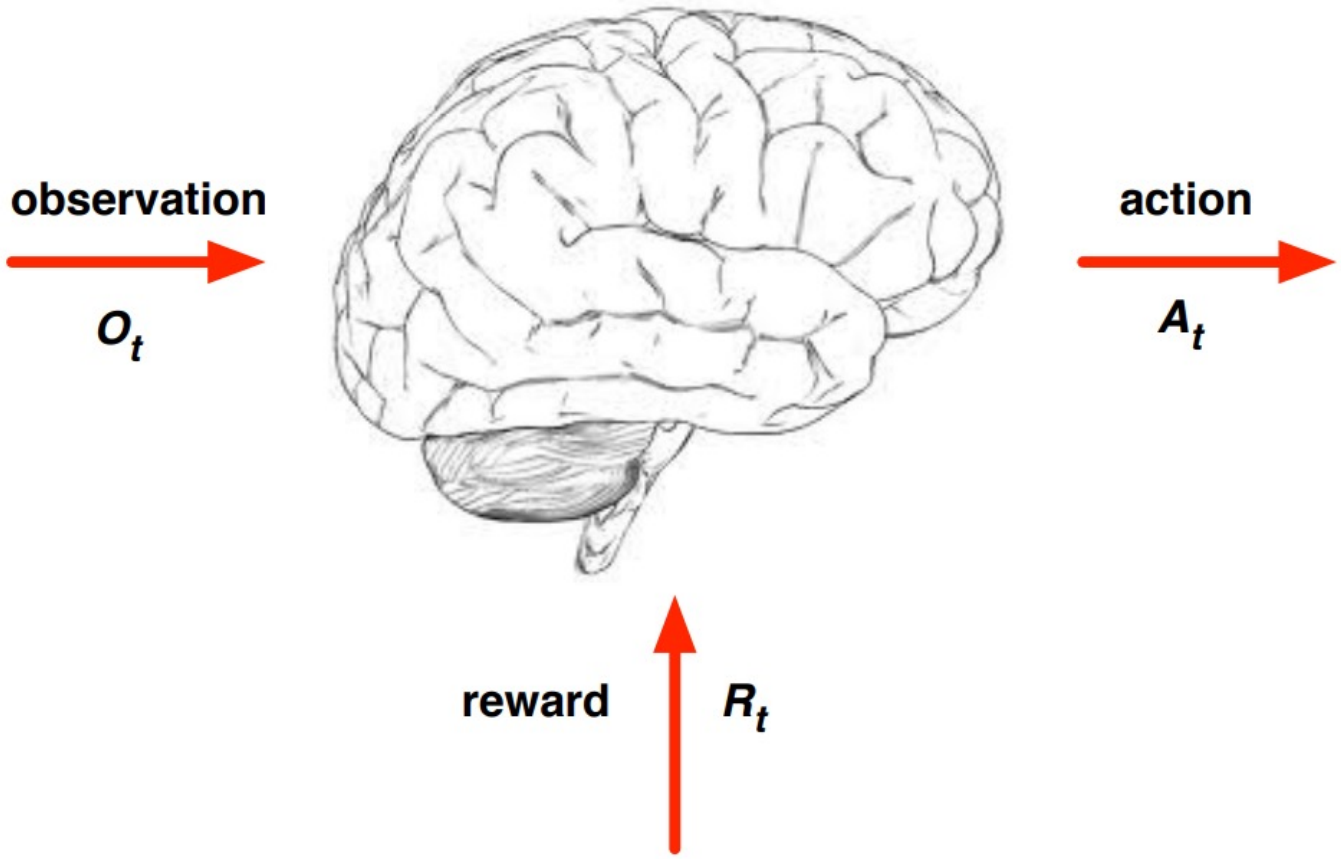
# Esempi di ricompense

- Far camminare un robot umanoide
  - ricompensa positiva per aver eseguito la traiettoria desiderata
  - ricompensa negativa in caso di caduta
- Sconfiggere il campione del mondo di Backgammon
  - ricompensa positiva/negativa per aver vinto/perso una partita
- Controllare una centrale elettrica
  - ricompensa positiva per aver prodotto energia
  - ricompensa negativa per aver violato le misure di sicurezza
- Gestire un portafoglio di investimenti
  - ricompensa positiva per ogni € guadagnato
- Giocare ai videogiochi
  - ricompensa positiva/negativa per aver migliorato/peggiorato il punteggio
- Scoprire nuove molecole (pos. molecola sintetizzabile, neg. molecola tossica)

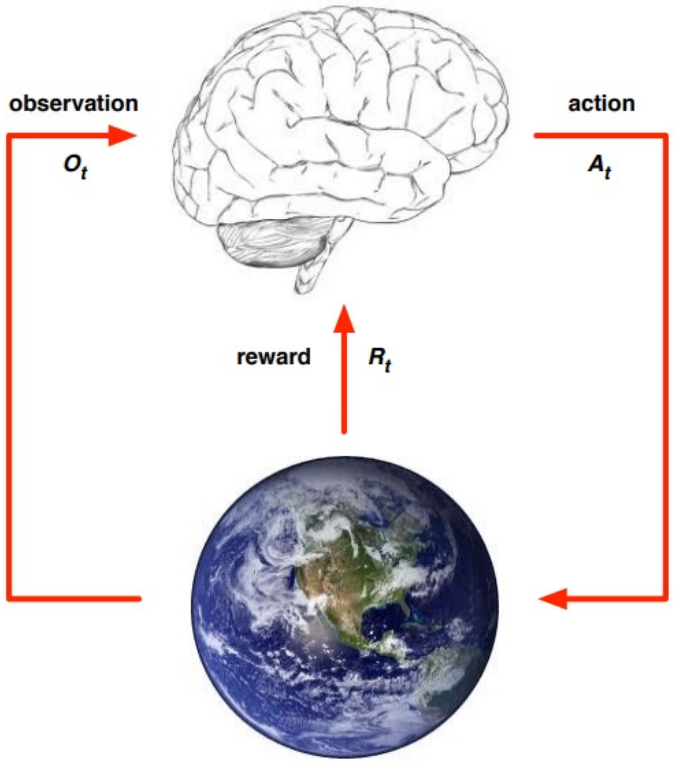
# Processo decisionale sequenziale

- Obiettivo: *selezionare le azioni al fine di massimizzare la ricompensa totale futura*
  - Le azioni possono avere conseguenze a lungo termine
  - La ricompensa potrebbe essere ritardata
  - Può essere preferibile sacrificare una ricompensa immediata per ottenere una maggiore ricompensa a lungo termine
- Esempi:
  - Un investimento finanziario (può richiedere mesi per maturare)
  - Rifornimento di carburante a un elicottero (potrebbe evitare un incidente dopo diverse ore)
  - Bloccare le mosse dell'avversario (potrebbe aiutare a vincere molte mosse da ora)

# Agente ed Ambiente



# Agente ed ambiente (2)




- A ciascun step *t* l'agente:
  - *Esegue l'azione* $A_t$
  - *Riceve l'osservazione* $O_t$
  - *Riceve la ricompensa* $R_t$
- L'ambiente:
  - *Riceve l'azione* $A_t$
  - *Produce l'osservazione* $O_{t+1}$
  - *Produce la ricompensa* $R_{t+1}$
- *t* viene incrementato

# Storia e Stato

- La *storia* è la sequenza di osservazioni, azioni, ricompense

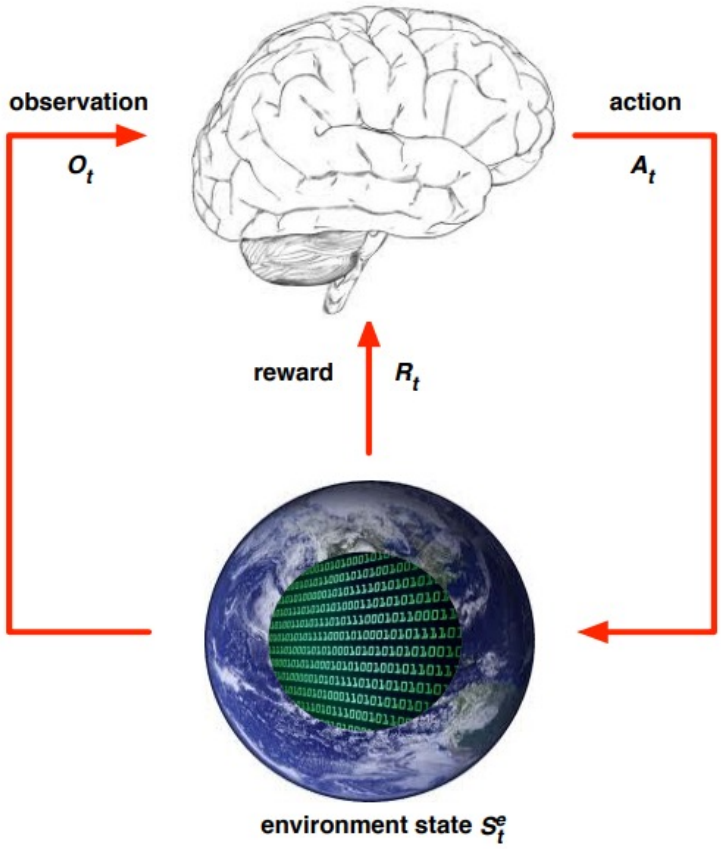
$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$$

  cioè tutte le variabili osservabili fino al tempo $t$

- La storia influenza ciò che accade successivamente:
  - Le azioni eseguite dall'agente
  - Le osservazioni/ricompense prodotte dall'ambiente
- Lo *stato* è l'informazione utilizzata per determinare ciò che accade successivamente
- Formalmente, lo stato $S_t$ è una funzione della storia:

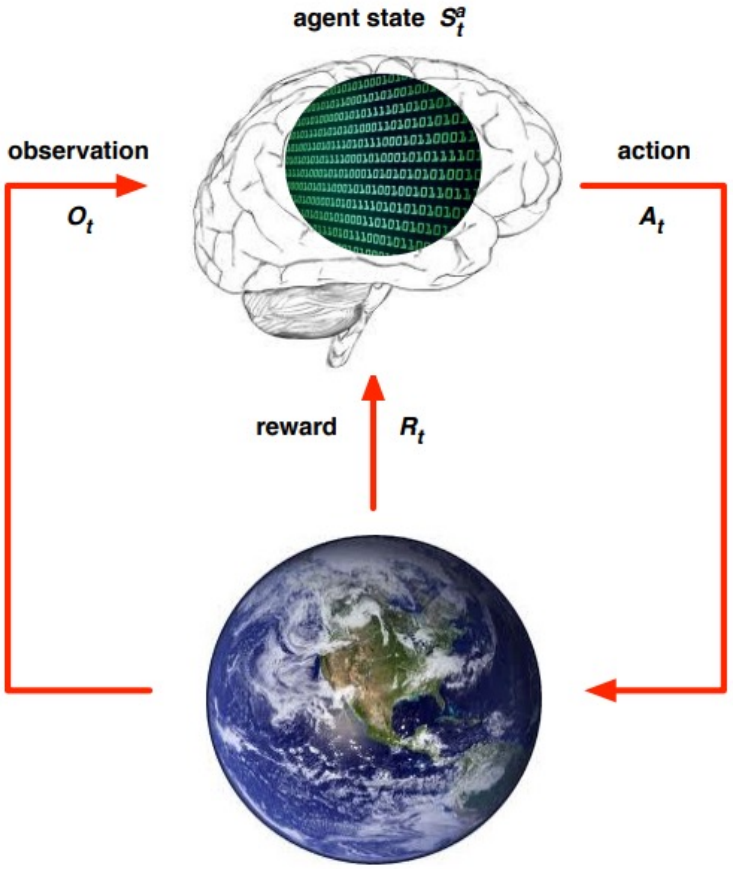
$$S_t = f(H_t)$$

# Stato dell'ambiente




- Lo stato dell'ambiente $S_t^e$ è la rappresentazione privata dell'ambiente *e* al tempo *t*
  - Qualsiasi informazione usata dall'ambiente per scegliere la prossima osservazione/ricompensa
- Di solito, lo stato dell'ambiente non è visibile all'agente
- Anche se $S_t^e$ è visibile, potrebbe contenere informazioni irrilevanti

# Stato dell'agente




- Lo stato dell'agente $S_t^a$ è la rappresentazione interna dell'agente *a*
  - Qualsiasi informazione usata dall'agente per scegliere l'azione successiva
- Questa è l'informazione utilizzata dagli algoritmi di RL
- Può essere qualsiasi funzione della storia:

$$S_t^a = f(H_t)$$

# Stato delle informazioni

- Uno stato delle informazioni (a.k.a. stato di Markov) contiene tutte le informazioni utili della storia
- Uno stato $S_t$ è detto di Markov se e solo se

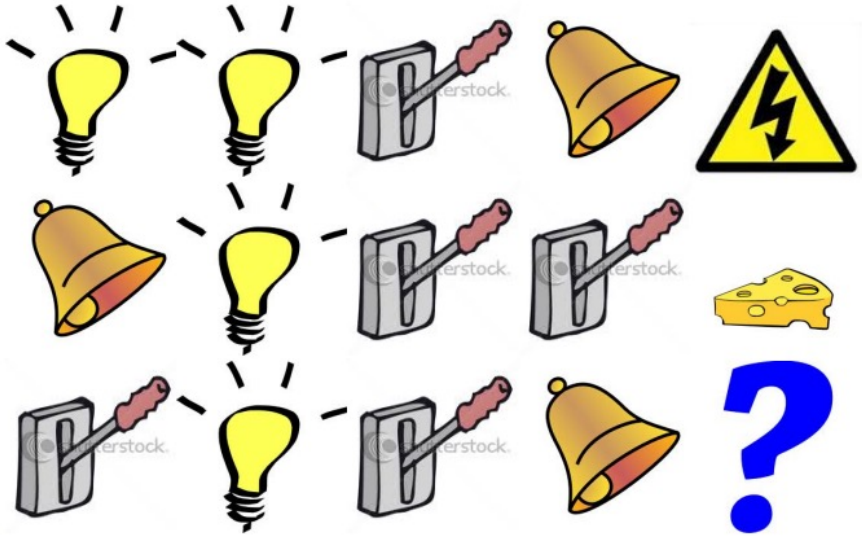
$$P[S_{t+1} \mid S_1, \ldots, S_t] = P[S_{t+1} \mid S_t]$$

«*Dato il presente, il futuro è indipendente dal passato*»

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

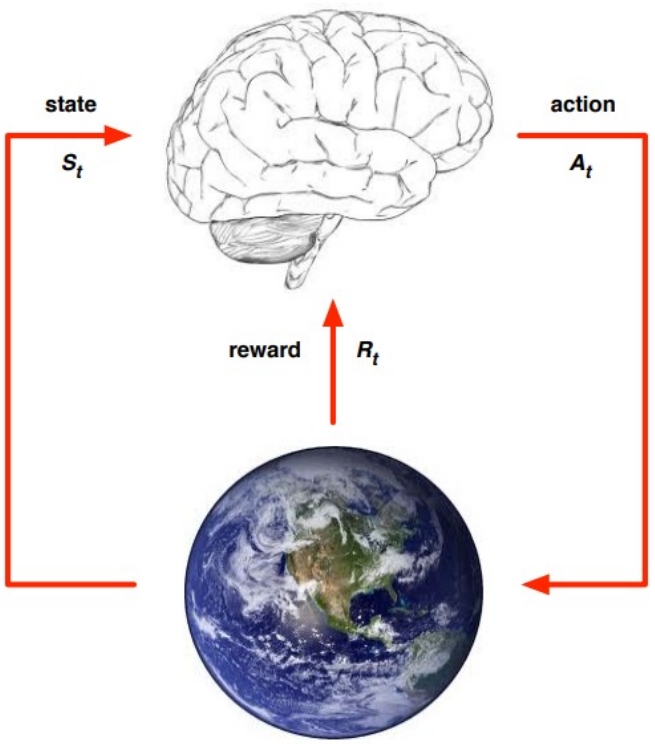
- Una volta che lo stato è noto, la storia può non essere considerata
- Lo stato è una statistica sufficiente del futuro
- Lo stato dell'ambiente $S_t^e$ e la storia $H_t$ sono di Markov

# Esempio


- Cosa succede se lo stato dell'agente = ultimi 3 elementi della sequenza?
- E se lo stato dell'agente = numero di luci, campanelli e leve?
- E se lo stato dell'agente = sequenza completa?

# Ambienti completamente osservabili




- Osservabilità completa: l'agente osserva direttamente lo stato dell'ambiente

$$O_t = S_t^a = S_t^e$$

- Stato dell'agente = Stato dell'ambiente = Stato delle informazioni
- Formalmente, questo è definito Processo Decisionale di Markov (MDP)

# Ambienti parzialmente osservabili

- Osservabilità parziale: l'agente osserva indirettamente l'ambiente
  - Un agente di trading osserva solo i prezzi correnti
  - Un agente che gioca a poker osserva solo le carte sul tavolo
- In questo caso Stato dell'agente ≠ Stato dell'ambiente
- Formalmente, questo è definito Processo Decisionale di Markov Parzialmente Osservabile (POMDP)
- L'agente deve costruire la propria rappresentazione dello stato $S_t^a$, ad esempio
  - Storia completa: $S_t^a = H_t$
  - Credenze sullo stato dell'ambiente: $S_t^a = (\mathbb{P}[S_t^e = s^1], \ldots, \mathbb{P}[S_t^e = s^n])$
  - Rete neurale ricorrente: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

# Componenti di un agente basato su Reinforcement Learning

# Componenti principali di un agente basato su RL

- Un agente basato su RL include una o più delle seguenti componenti:
  - Policy: funzione di comportamento dell'agente
  - Value function: esprime quanto è valido ogni stato e/o azione
  - Modello: rappresentazione dell'ambiente ad opera dell'agente

# Policy

- Una policy $\pi$ esprime il comportamento dell'agente
- Stabilisce l'azione da eseguire in base allo stato in cui si trova l'agente, ad esempio
  - Policy deterministica: $a = \pi(s)$
  - Policy stocastica: $\pi(a|s) = P[A_t = a | S_t = s]$

# Value function

- Una value function è una previsione della ricompensa futura
- Viene utilizzata per valutare la validità degli stati e quindi per la scelta delle azioni da eseguire, ad esempio

$$v_\pi(s) = \mathbb{E}_\pi \left[ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s \right]$$

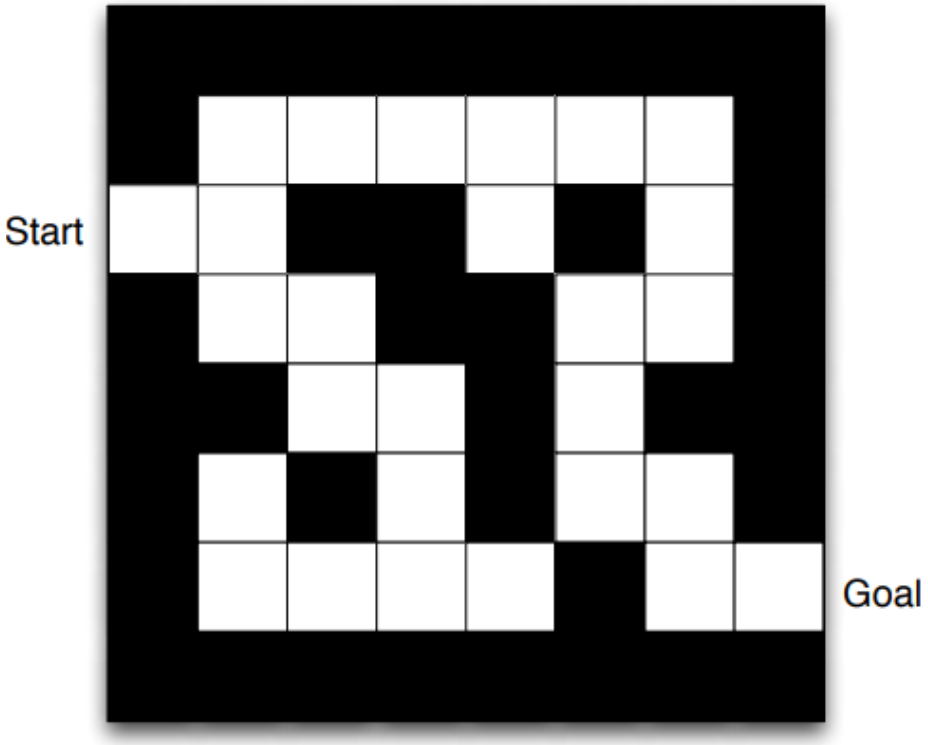
- Ricompensa futura attesa (scontata) in base alla politica $\pi$ dallo stato $s$
- Uno stato $s$ potrebbe avere sempre una ricompensa bassa ma avere un valore $v_\pi(s)$ alto (gli stati seguenti hanno una ricompensa alta)

# Modello

- Un modello predice ciò che l'ambiente farà in seguito
- $\mathcal{P}$ predice il prossimo stato
- $\mathcal{R}$ predice la ricompensa successiva (immediata), ad esempio

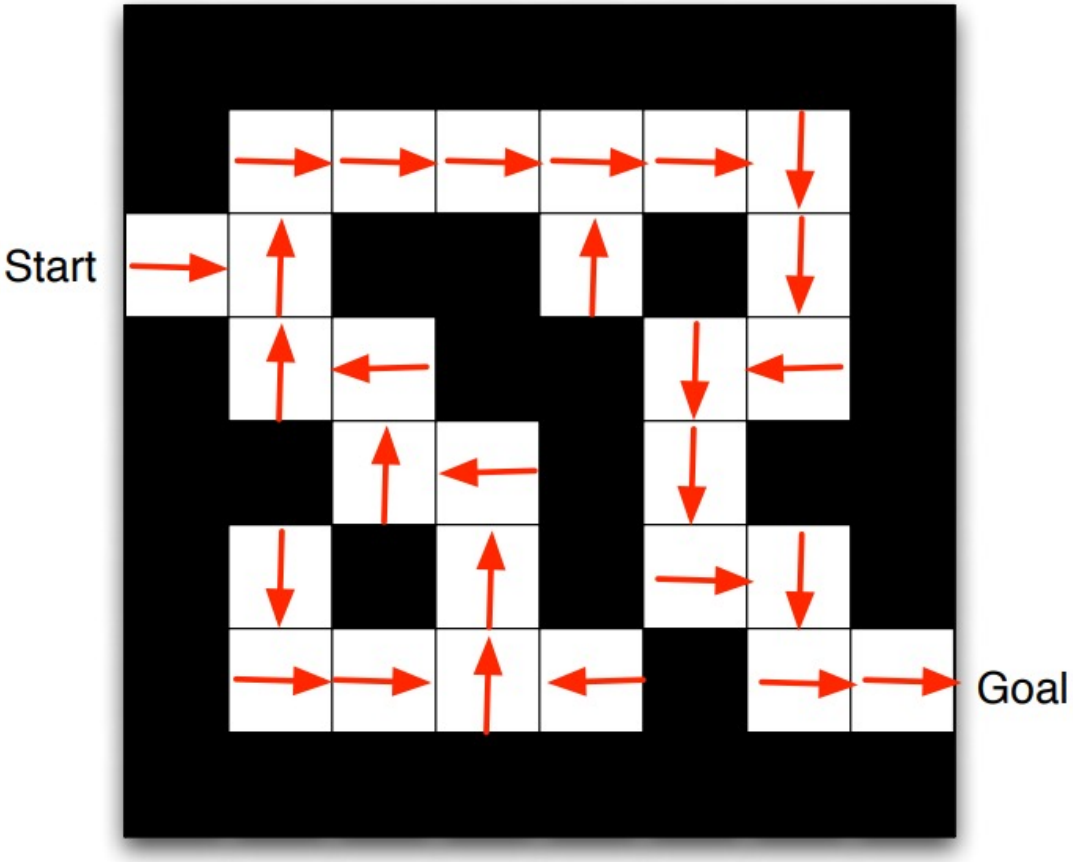
$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

# Esempio Labirinto



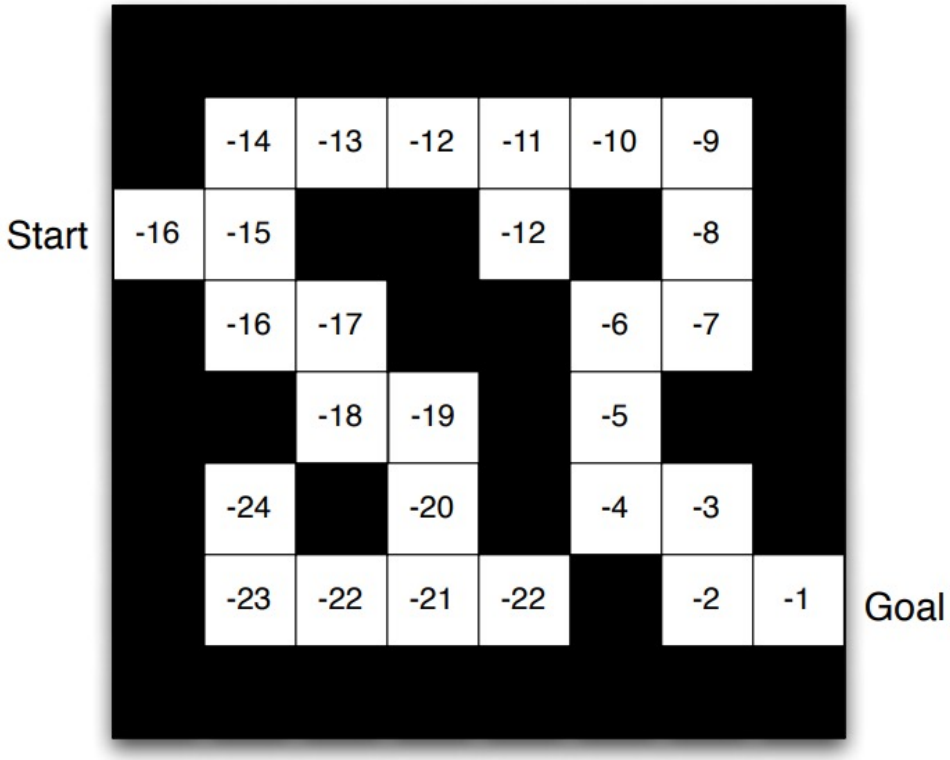

- Ricompense: -1 per time-step
- Azioni: N, E, S, O
- Stati: posizione dell'agente

# Esempio Maze: Policy



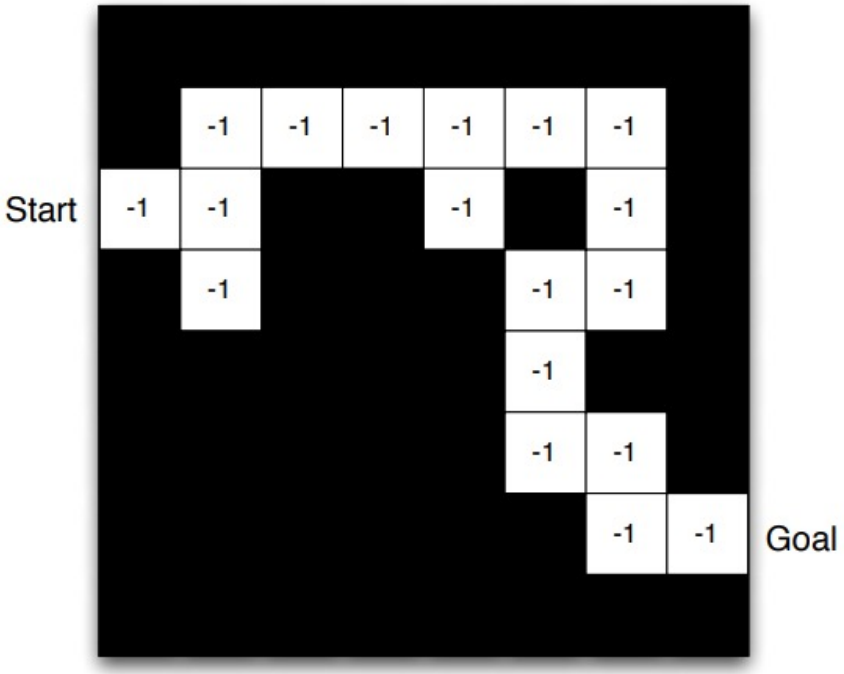

- Le frecce rappresentano la policy π(s) per ogni stato s

# Esempio Maze: Value Function




- I numeri rappresentano il valore $v_\pi(s)$ di ciascuno stato *s*
- *Tempo previsto per raggiungere l'obiettivo*

# Esempio Maze: Modello




- L'agente può avere un modello interno (imperfetto) dell'ambiente
  - Come le azioni cambiano lo stato
  - Quanta ricompensa si ottiene da ogni stato

- Il layout a griglia rappresenta il modello di transizione $\mathcal{P}^a_{ss'}$
- I numeri rappresentano la ricompensa immediata $\mathcal{R}^a_s$ da ciascuno stato s

# Apprendere le componenti degli agenti

- Tutte le componenti sono funzioni
  - Policies: $\pi : \mathcal{S} \rightarrow \mathcal{A}$ (or to probabilities over $\mathcal{A}$)
  - Value functions: $v : \mathcal{S} \rightarrow \mathbb{R}$
  - Models: $m : \mathcal{S} \rightarrow \mathcal{S}$ and/or $r : \mathcal{S} \rightarrow \mathbb{R}$
  - State update: $u : \mathcal{S} \times \mathcal{O} \rightarrow \mathcal{S}$
- Ad es., possiamo usare reti neurali o tecniche di deep learning per apprenderle
- NB. Spesso non valgono le assunzioni del supervised learning (stazionarietà, iid)

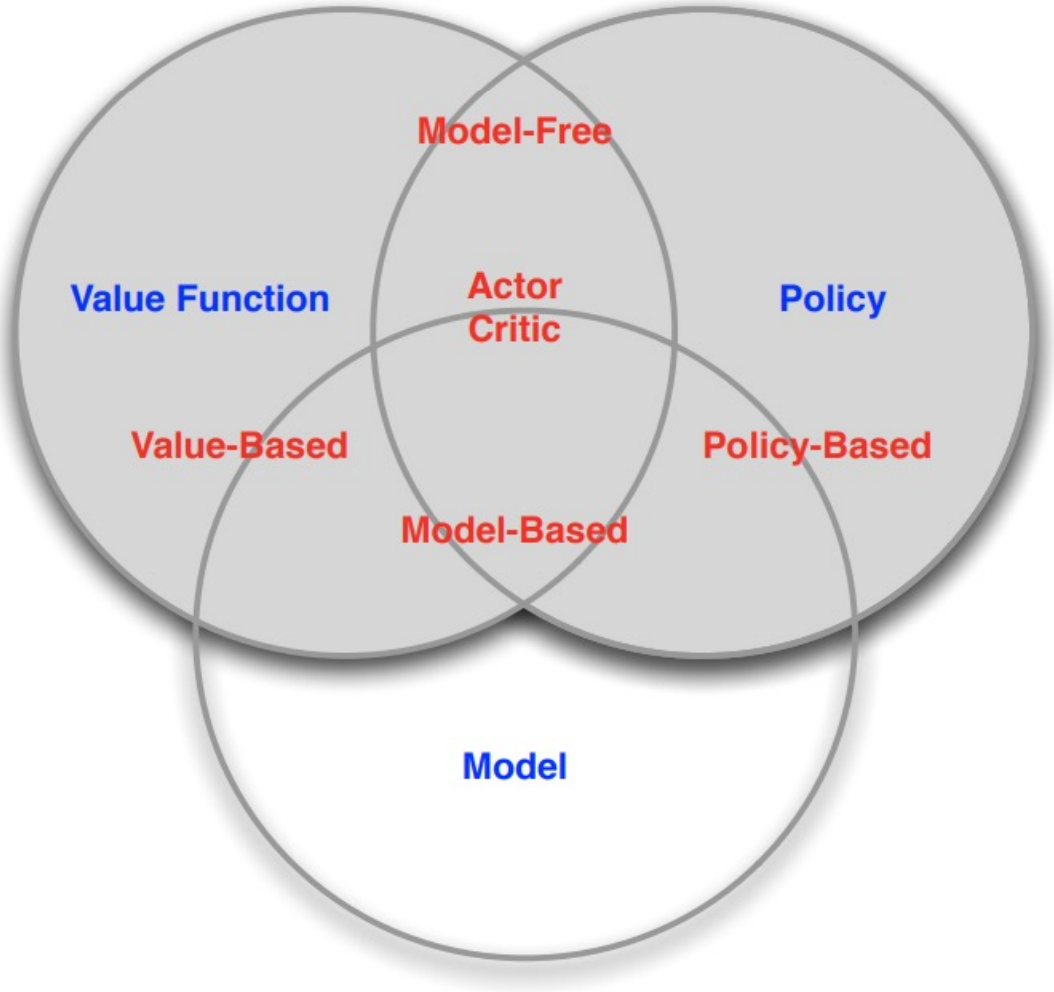
# Categorizzazione degli agenti basati su RL

- Value-based
  - Non utilizzano alcuna Policy
  - Value Function
- Policy-based
  - Policy
  - Non utilizzano alcuna Value Function
- Actor Critic
  - Policy
  - Value Function

# Categorizzazione degli agenti basati su RL (2)

- Model-free
  - Policy e/o Value Function
  - Non utilizzano alcun Modello
- Model-based
  - Policy e/o Value Function
  - Model

# Tassonomia degli agenti basati su RL

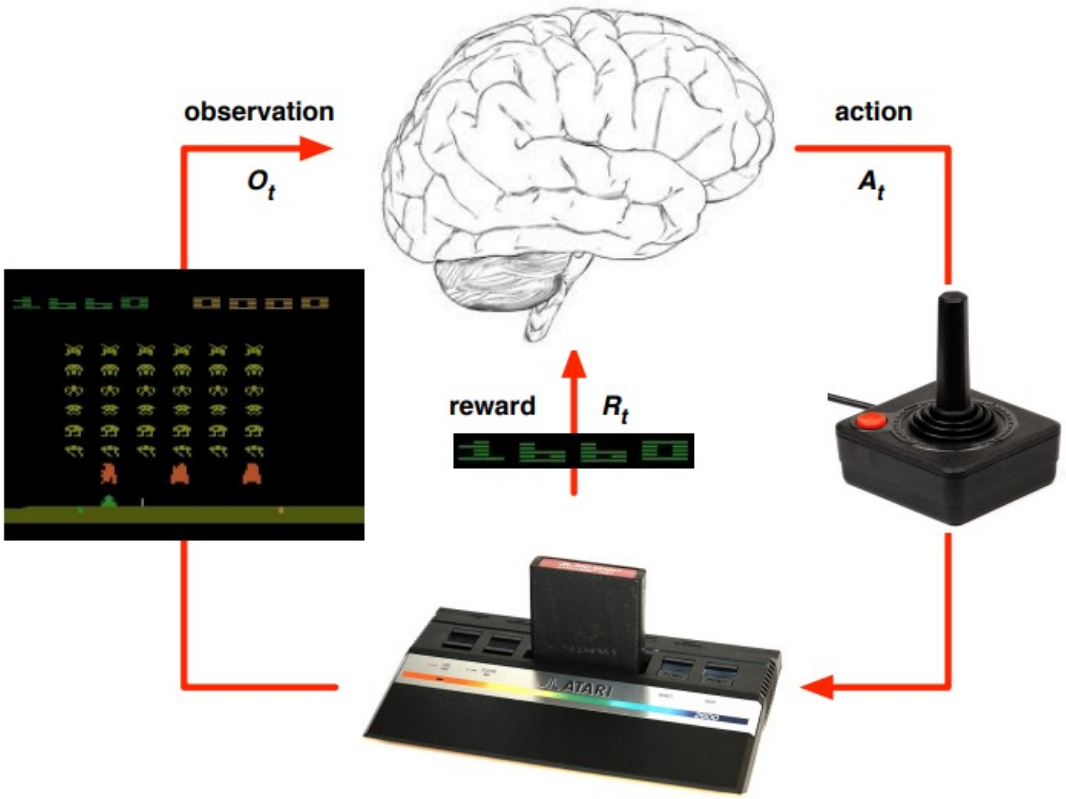


# Learning e Planning

Ci sono due problematiche fondamentali in un processo decisionale sequenziale

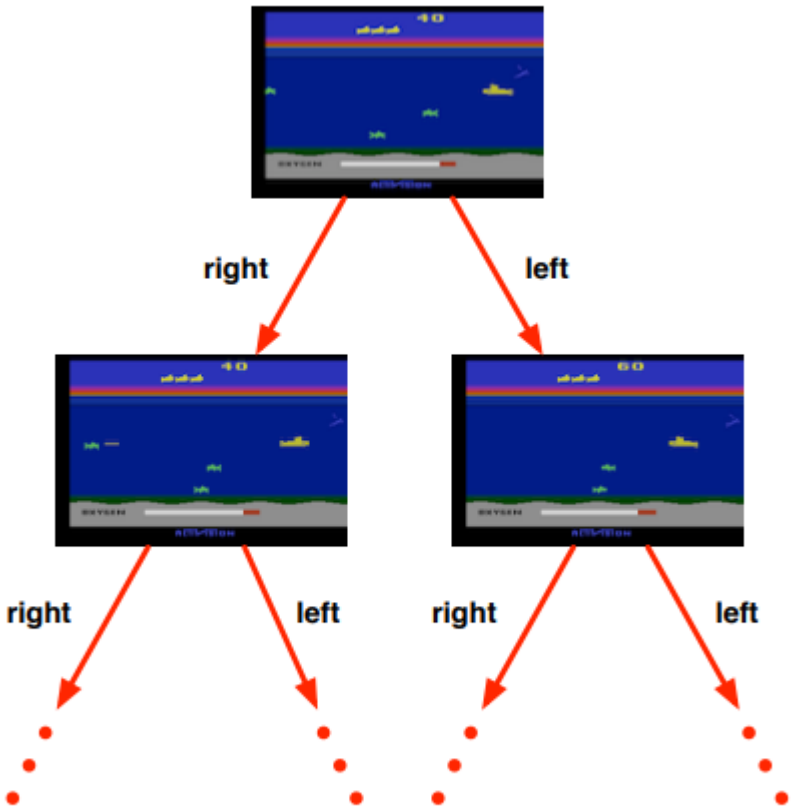
- Reinforcement Learning:
  - L'ambiente è inizialmente sconosciuto
  - L'agente interagisce con l'ambiente
  - L'agente migliora la sua policy
- Planning:
  - Un modello dell'ambiente è noto
  - L'agente esegue le computazioni con il suo modello (senza alcuna interazione esterna)
  - L'agente migliora la sua policy

# Esempio Atari: Reinforcement Learning




- Le regole del gioco sono sconosciute
- L'apprendimento è basato sull'esperienza ottenuta dall'interazione con il gioco
- Bisogna scegliere le azioni da eseguire col joystick, vedere i pixel e i punteggi

# Esempio Atari: Planning




- Le regole del gioco sono note
- L'agente può interrogare l'emulatore
  - Esiste un modello perfetto per l'agente
- Se l'agente esegue un'azione *a* da uno stato *s*:
  - Quale è il prossimo stato?
  - Quale è il punteggio?
- Bisogna pianificare in anticipo per trovare una policy ottimale
  - Ad esempio, si può usare un albero di ricerca

# Exploration and Exploitation

- Il Reinforcement Learning è simile ad un apprendimento *trial-and-error*

- L'agente dovrebbe individuare una politica buona
  - Tramite l'esperienza che acquisisce interagendo con l'ambiente
  - Senza perdere troppa ricompensa lungo il percorso

# Exploration and Exploitation (2)

- L'*exploration* trova più informazioni sull'ambiente
- L'*exploitation* sfrutta le informazioni note per massimizzare la ricompensa
- Di solito questi due task hanno la stessa importanza

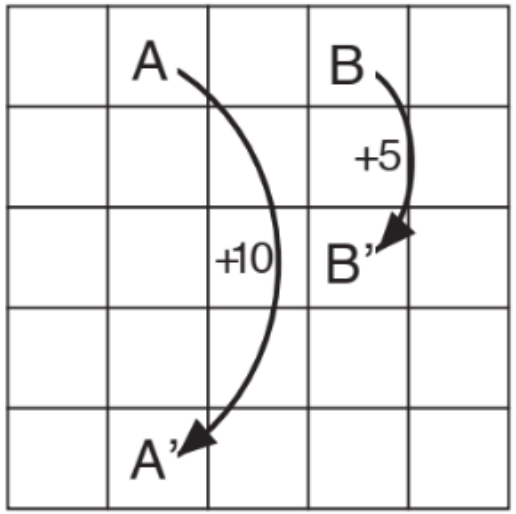
# Esempi

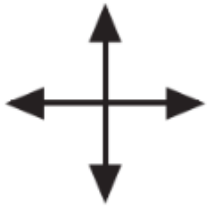
- Selezione di un ristorante
  - Exploitation: Vai al tuo ristorante preferito
  - Exploration: Prova un nuovo ristorante
- Banner pubblicitari online
  - Exploitation: Mostra la Pubblicità di maggior successo
  - Exploration: Mostra una pubblicità diversa
- Perforazioni petrolifere
  - Exploitation: Perforare nel punto più noto
  - Exploration: Perforare in una nuova posizione
- Game-playing
  - Exploitation: Esegui la mossa che ritieni migliore
  - Exploration: Esegui una mossa sperimentale

# Prediction e Control

- Prediction: valutare il futuro
  - La policy è nota
- Control: ottimizzare il futuro
  - Trova la miglior policy

# Esempio Gridworld: Prediction



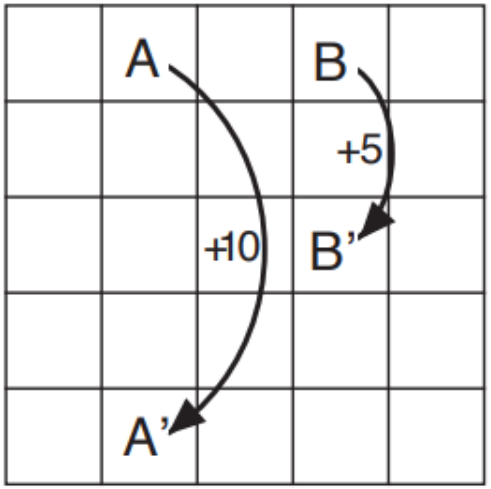



Actions

| | | | | |
|---|---|---|---|---|
| 3.3 | 8.8 | 4.4 | 5.3 | 1.5 |
| 1.5 | 3.0 | 2.3 | 1.9 | 0.5 |
| 0.1 | 0.7 | 0.7 | 0.4 | -0.4 |
| -1.0 | -0.4 | -0.4 | -0.6 | -1.2 |
| -1.9 | -1.3 | -1.2 | -1.4 | -2.0 |

- Quale è la value function per la policy casuale uniforme?

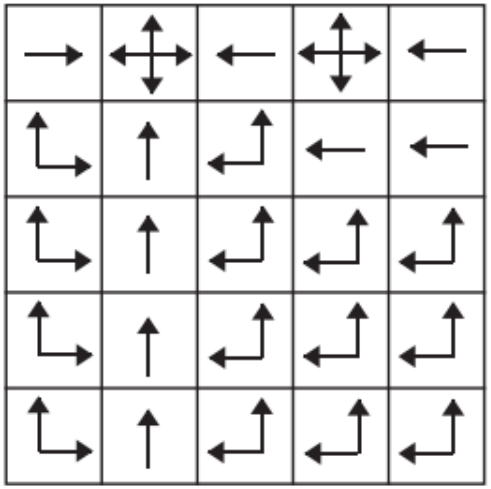
# Esempio Gridworld: Control




a) gridworld

| | | | | |
|---|---|---|---|---|
| 22.0 | 24.4 | 22.0 | 19.4 | 17.5 |
| 19.8 | 22.0 | 19.8 | 17.8 | 16.0 |
| 17.8 | 19.8 | 17.8 | 16.0 | 14.4 |
| 16.0 | 17.8 | 16.0 | 14.4 | 13.0 |
| 14.4 | 16.0 | 14.4 | 13.0 | 11.7 |

b) $v_*$



c) $\pi_*$

- Quale è la value function ottimale tra tutte le possibili policy?
- Quale è la policy ottimale?