

Soft17

An AI Agent for Blackjack

Progetto di Fondamenti di
Intelligenza Artificiale

Il team



Abbatiello Simone

mat. 0512119659



Nappi Vincenzo

mat. 0512119813



Niemiec Francesco

mat. 051211899



BLACKJACK

In breve



Il giocatore parte con due carte scoperte ed osserva una singola carta del banco, sceglie tra pescare o fermarsi.

Vince se la somma è 21, il banco fa più di 21 o meno di lui, altrimenti perde.

CASINO

WIN



Molta teoria ma...



Journal of Statistical Planning and
Inference

Volume 33, Issue 3, December 1992, Pages 295-309



Machine learning for optimal blackjack counting strategies

S. Yakowitz , M. Kollar

4,809

Views

1

CrossRef

citations to date

0

Altmetric



Free access

Book Reviews

Mathematics of The Big Four Casino Table Games: Blackjack, Baccarat, Craps, & Roulette

Stan Lipovetsky

Pages 613-614 | Published online: 03 Nov 2023

Cite this article

<https://doi.org/10.1080/00401706.2023.2262898>



Article

The Optimum Strategy in Blackjack

Roger R. Baldwin, Wilbert E. Cantey, Herbert Maisel & James P. McDermott

Pages 429-439 | Published online: 11 Apr 2012

Cite this article

Computer Science > Artificial Intelligence

[Submitted on 9 Aug 2023]

Variations on the Reinforcement Learning performance of Blackjack

Avish Buramdayal, Tim Gebbie

Blackjack or "21" is a popular card-based game of chance and skill. The objective of the game is to win by obtaining a hand total higher than the dealer's without exceeding 21. The ideal blackjack strategy will maximize financial return in the long run while avoiding gambler's ruin. The stochastic environment and inherent reward structure of blackjack presents an appealing problem to better understand reinforcement learning agents in the presence of environment variations. Here we consider a q-learning solution for optimal play and investigate the rate of learning convergence of the algorithm as a function of deck size. A blackjack simulator allowing for universal blackjack rules is also implemented to demonstrate the extent to which a card counter perfectly using the basic strategy and hi-lo system can bring the house to bankruptcy and how environment variations impact this outcome. The novelty of our work is to place this conceptual understanding of the impact of deck size in the context of learning agent convergence.

...troppa pratica?

≡ Menu

🔍 Cerca

la Repubblica **50**

ABBONATI



Accedi



Un terzo dei ragazzi tra 15 e 19 anni che usa i videogame gioca d'azzardo

di Valeria Pini



A sostenerlo uno studio appena pubblicato su Lancet Public Health. Le conseguenze della ludopatia sul cervello degli adolescenti

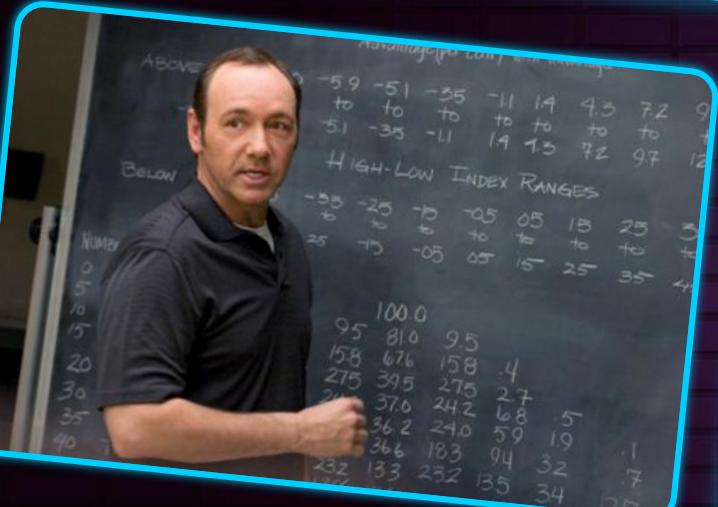
23 DICEMBRE 2025 ALLE 00:00

3 MINUTI DI LETTURA

Il nostro obiettivo



creare un agente che sia in grado di
giocare a Blackjack in maniera efficiente
ed autonoma



Due approcci



Apprendimento supervisionato

imparare da dati etichettati per fare previsioni su nuovi elementi



Reinforcement Learning

imparare a prendere decisioni interagendo con un ambiente e ricevendo ricompense o penalità



Quattro modalità

01

Decision Tree

creare un albero in cui i nodi sono sottoinsiemi di caratteristiche del problema e gli archi rappresentano le decisioni.

02

Random Forest

insieme di molti alberi decisionali che votano insieme per fare previsioni più accurate e stabili.

03

Q-Learning

imparare la miglior azione indipendentemente dalla politica eseguita

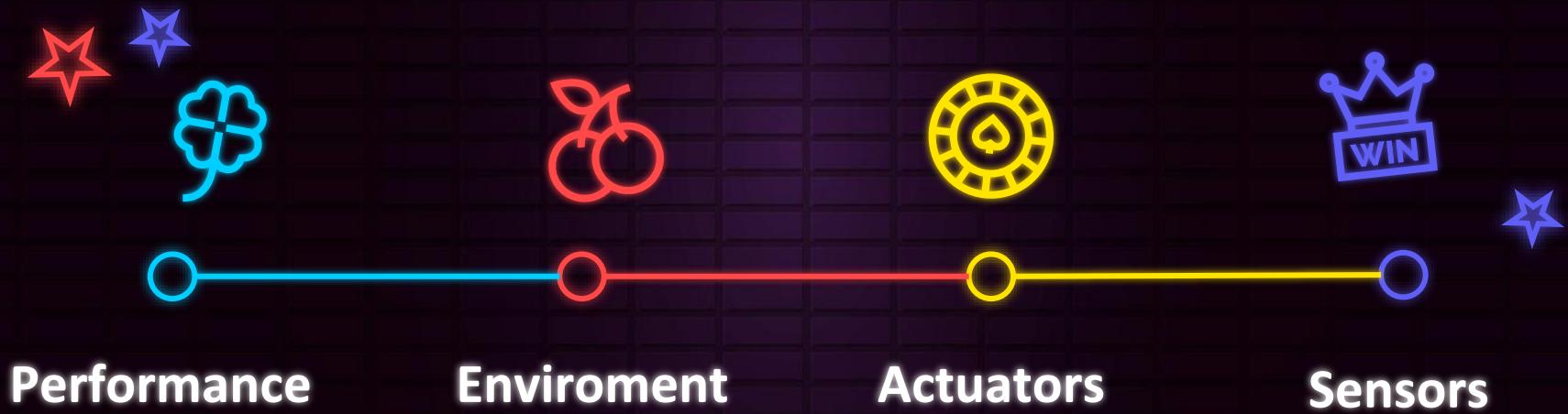
04

SARSA

aggiornare i valori in base all'azione realmente eseguita dalla politica corrente.



Specifica PEAS



Caratteristiche dell'ambiente

Parzialmente osservabile

sono noti solo la propria mano e la carta scoperta del dealer

Stocastico

l'ordine delle carte è pseudocasuale

Sequenziale

ogni azione influenza l'esito finale della mano



Statico

durante la decisione l'ambiente non cambia

Discreto

seppur grande, il numero di stati ed azioni è finito

Singolo agente

il dealer segue una strategia prefissata e non è considerato un agente decisionale.





01 - 02

Apprendimento
Supervisionato

Gli step di data engineering



Esplorazione

visualizzare il dataset e prendere
confidenza con le sue
caratteristiche

Data Cleaning

affrontare dati mancanti o
rumorosi (non applicato)

Feature Scaling

normalizzare valori per
non confondere il
modello



Data Balancing

bilanciamento delle
feature nel dataset (non
applicato)

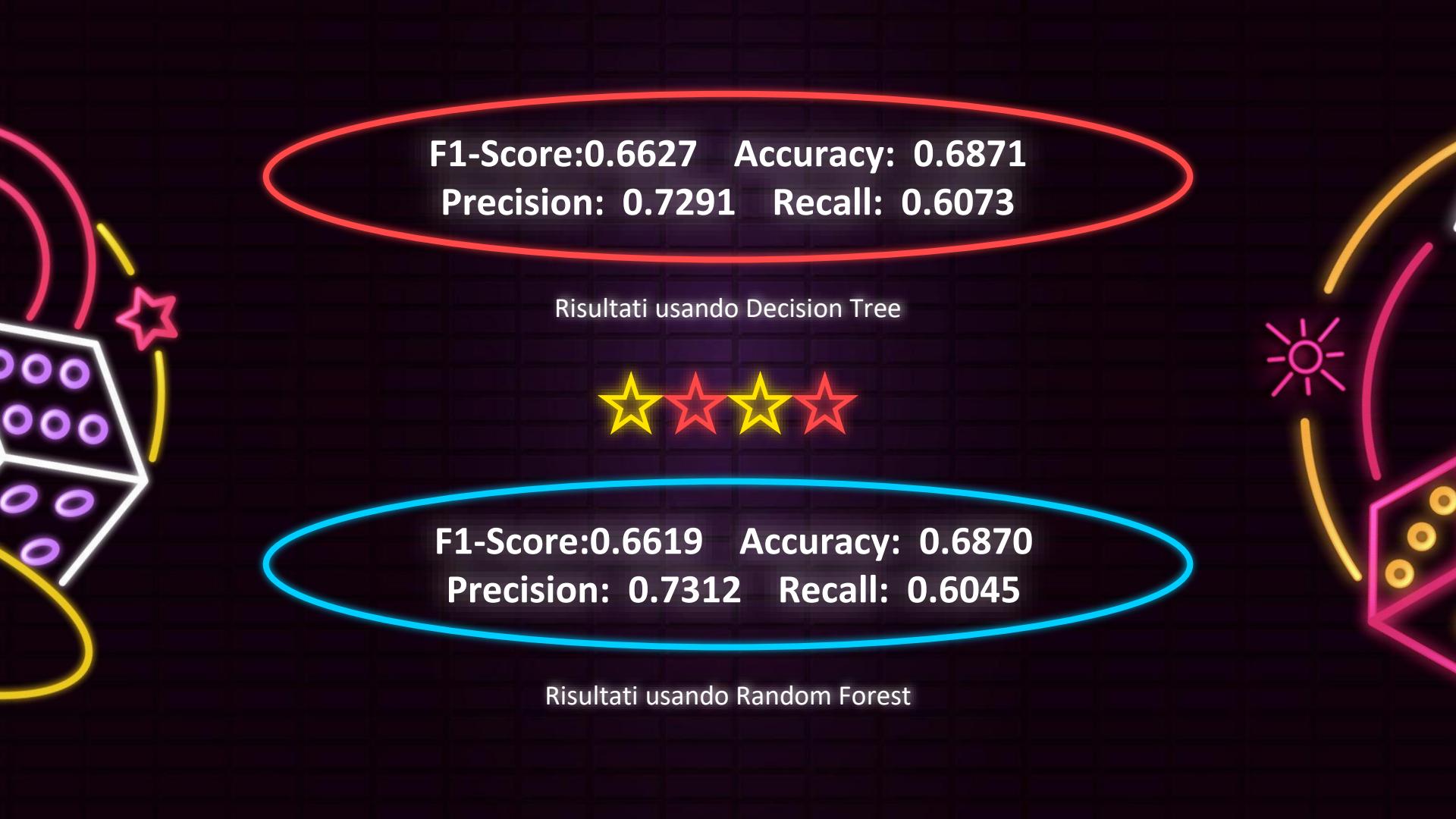
Feature Construction

creazione di nuove
colonne rilevanti

Feature Selection

eliminazione di feature di
bassa importanza





**F1-Score: 0.6627 Accuracy: 0.6871
Precision: 0.7291 Recall: 0.6073**

Risultati usando Decision Tree



**F1-Score: 0.6619 Accuracy: 0.6870
Precision: 0.7312 Recall: 0.6045**

Risultati usando Random Forest

Conclusioni tratte



**Feedback
intermedio**

potrebbe essere utile

**Mancanza di
sequenzialità**

non particolarmente
catturata



Falsi negativi

troppi per un utilizzo
utile

**Similarità dei
modelli**

Più complicato non
comporta necessariamente
miglioramenti



03 - 04

REINFORCEMENT LEARNING



Off-Policy vs On-policy



Q-Learning



impara la miglior azione da eseguire in ogni stato indipendentemente dalla policy esplorativa, aggiorna la Q-Table usando l'azione con la miglior ricompensa, nel nostro caso ciò comporta un atteggiamento aggressivo.



SARSA

aggiorna la Q-Table basandosi sull'azione migliore scelta dalla politica corrente, considerandola come quella ottimale, nel nostro caso l'atteggiamento sarà molto conservativo.



L'ALGORITMO SARSA



Stato

compresso in una tupla del tipo
`s = (player_sum, is_soft,
dealer_showing)`

Scelta azione

con probabilità ϵ scelgo
un'azione casuale,
altrimenti quella con valore Q
più alto

Interazione con l'ambiente

si ottiene il nuovo stato
ed una ricompensa



Decremento

riduco il valore di ϵ e
ripetto

Aggiornamento

si aggiorna la
Q-Table

Scelta prossima azione

quando non terminale, si sceglie
subito la prossima azione con la
stessa politica

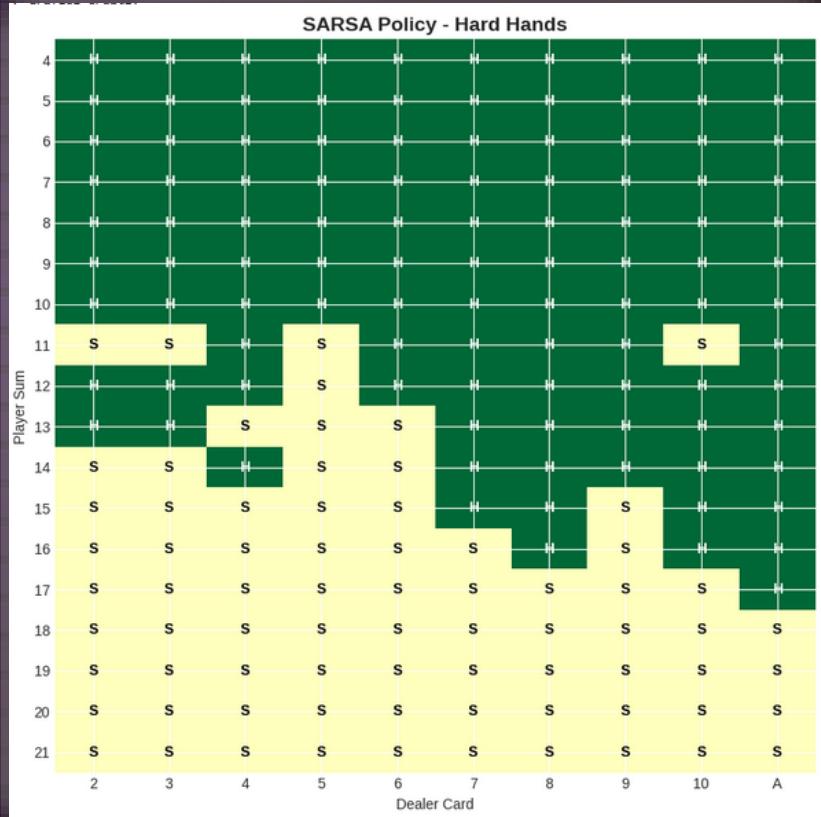
Heatmap binaria - SARSA

Ogni cella della mappa corrisponde a uno stato del tipo:

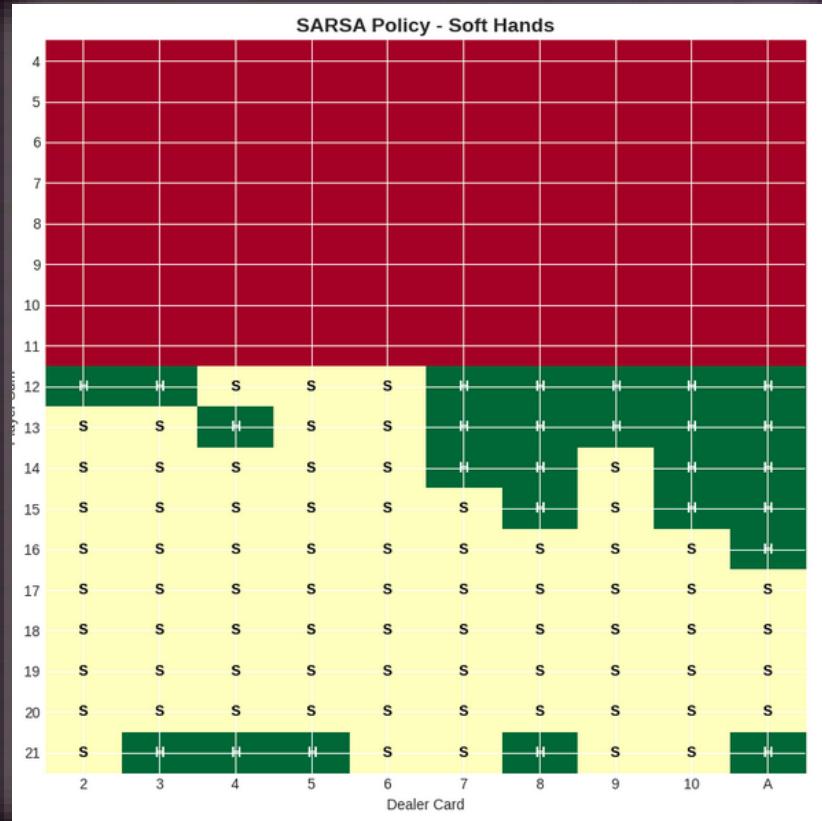
$$s = (\text{player_sum}, \text{is_soft} = 0, \text{dealer_showing})$$

e il colore (o il simbolo H/S) indica l'azione ritenuta ottimale dall'agente in quello stato, ovvero Hit o Stand.

Qui si fa esplicito riferimento ai casi in cui un asso vale 11.



HeatMap binaria - SARSA



Qui invece al
caso in cui ogni
asso vale 1

L'agente usa ciò che già sa per massimizzare la ricompensa. Quindi prende l'azione con il **valore Q più alto** nello stato corrente.

- Exploitation





L'agente prova azioni **non necessariamente ottimali** per imparare di più sull'ambiente. Serve per scoprire stati e azioni che potrebbero essere migliori di quelle conosciute.

- Exploration

Decayed Epsilon Greedy



Inizio

Inizialmente preferiamo l'esplorazione (epsilon = 1)



Fasi intermedie

Man mano diminuiamo il valore di epsilon, bilanciando exploration ed exploitation



Fasi finali

Diamo più importanza alle conoscenze che all'esplorazione.

Win Rate

Q-Learning (Offline)

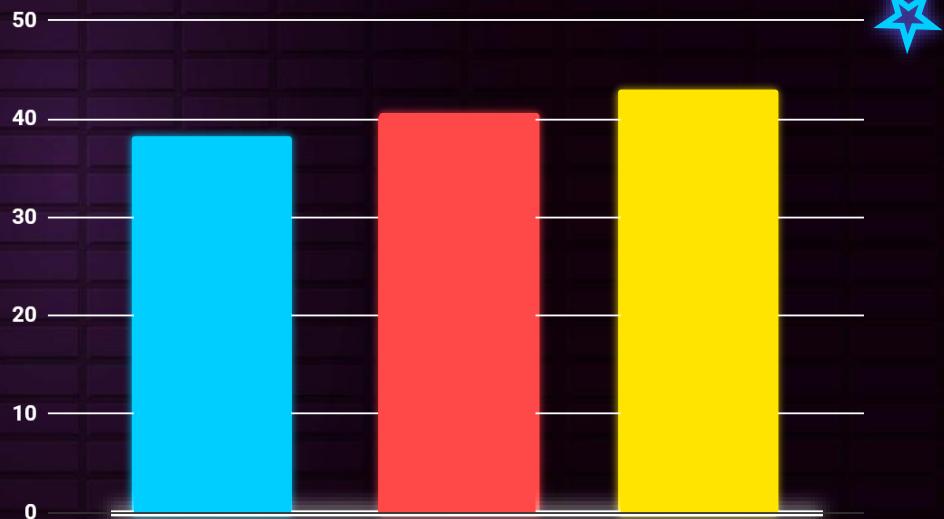
38.14
%

Q-Learning (Online)

40.29
%

SARSA (Online)

42.67
%



la strategia ottimale di Blackjack ha accuracy stimata attorno al 42%-
43%.

Una strategia totalmente randomica
Invece raggiunge il 28%

Considerazioni finali

Risultato ottimo

l'agente riesce ad imparare strategie molto simili a quella ottimale, senza esserne esposto direttamente



Esperienza acquisita

grazie al progetto abbiamo ottenuto esperienza diretta con gli algoritmi di reinforcement learning e migliorato le nostre conoscenze sull'apprendimento supervisionato

Miglioramenti futuri

Sperimentando con varianti degli algoritmi proposti è possibile ottenere miglioramenti, ma dato l'inerente vantaggio del banco questi non saranno considerevoli

Backup Slide

GridSearchCV

Usato nella prima pipeline, serve a trovare la miglior combinazione di iperparametri di un modello.

Le prova tutte le usando cross-validation.



Q-Table

Definizione

Tabella che associa a ogni stato tutte le possibili azioni e il loro valore atteso

Inizio

Inizialmente la tabella è inizializzata con tutti i valori 0

Al termine la Q-Table è composta da 280 stati

Stati finali

L'ALGORITMO Q-Learning



Stato

compresso in una tupla del tipo
`s = (player_sum, is_soft,
dealer_showing)`

Scelta azione

con probabilità ϵ scelgo
un'azione casuale,
altrimenti quella con valore Q
più alto

Interazione con l'ambiente

si ottiene il nuovo stato
ed una ricompensa



Decremento

riduco il valore di ϵ e
riproto

Ripeto nello stesso episodio

aggiorno $s = s'$ ripetendo finché l'episodio
non termina.
Senza scegliere subito l'azione come in
SARSA

Aggiornamento

si aggiorna la
Q-Table, usando il **massimo
valore futuro**
indipendentemente dall'azione
scelta nel prossimo stato