UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

# Intelligenza Artificiale

## Value Function Approximation

a.a. 2022/2023

# Outline

- Introduzione
- Metodi incrementali
  - Stochastic Gradient Descent
  - Gradient TD Learning
  - Linear value approximation
- Metodi batch
  - Linear least squares
  - Experience replay & DQN

# Reinforcement Learning on the Scale

- Vogliamo utilizzare il reinforcement learning in problemi con uno spazio degli stati non banale
  - Backgammon: 1020 stati
  - Go: 10170 stati
  - Robot: spazio degli stati continuo
  - Ricerca di molecole: >$10^{60}$ stati
- Possiamo scalare i metodi model-free visti nelle scorse lezioni?

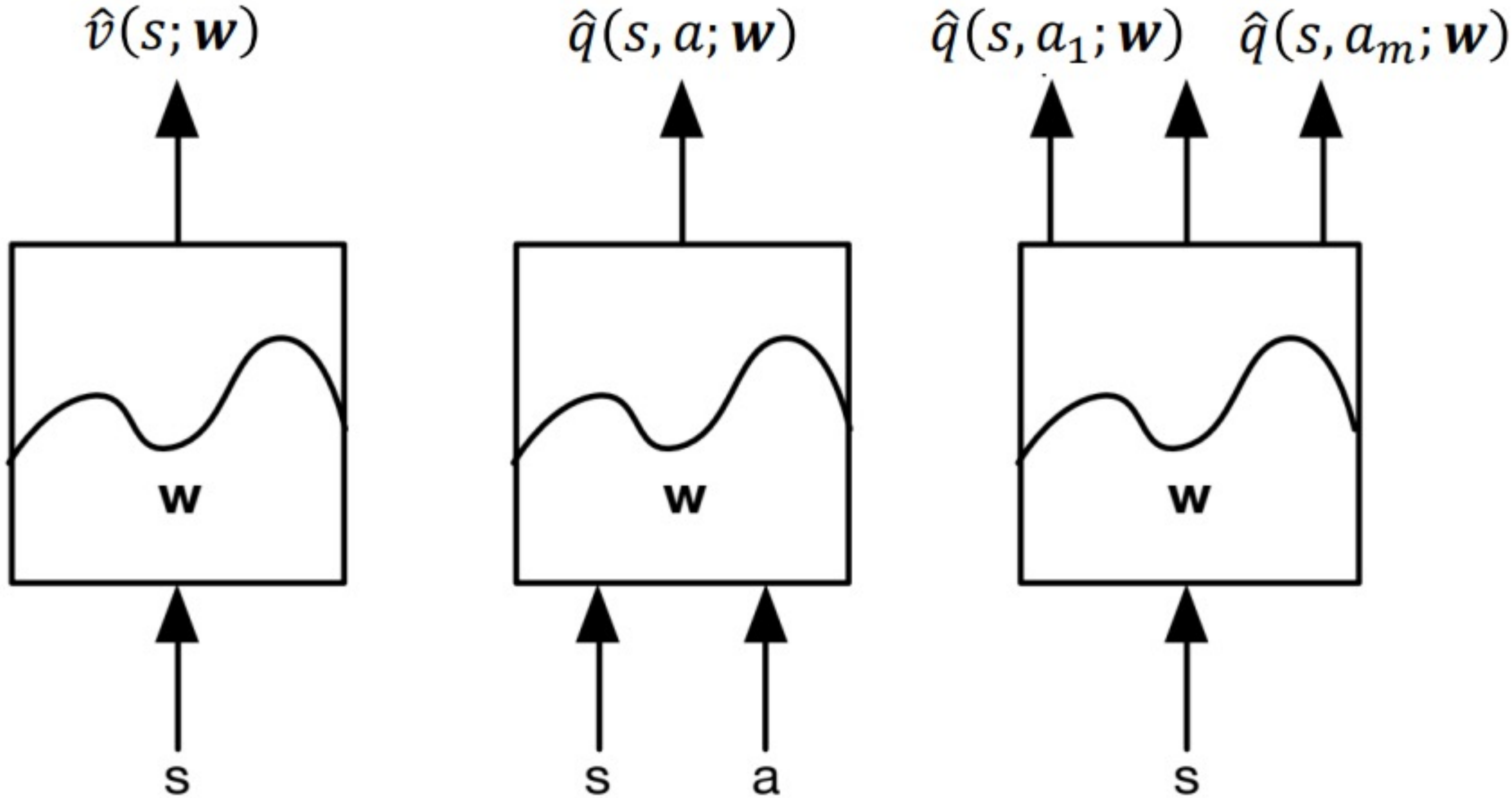
# Value Function Approximation

- Finora $V(s)/Q(s, a)$ = lookup table
  - Una entry per ogni stato *s* o coppia stato-azione *s*, *a*
  - MDP di grandi dimensioni $\Longrightarrow$ troppi stati e/o azioni da immagazzinare in memoria
  - Troppo lento per apprendere il valore di ogni singolo stato
  - Problemi di generalizzazione
- Il nuovo approccio
  - Stimare la value function con la approximation function

$$\hat{v}(s;\mathbf{w}) \approx v_\pi(s)$$
$$\hat{q}(s,a;\mathbf{w}) \approx q_\pi(s,a)$$

  - Generalizzare da stati osservati a stati non osservati
  - Aggiornare i parametri ***w*** utilizzando MC o TD learning

# Value Function Approximation - Approcci



# Quale Function Approximator?

- Combinazioni lineari di feature
- Rete neurale
- Decision tree
- Nearest neighbour
- Fourier / Wavelet bases
- …

# Quale Function Approximator?

- Combinazioni lineari di feature
- Rete neurale

Focus sui metodi differenziabili

- Decision tree
- Nearest neighbour
- Fourier / Wavelet bases
- …
- Inoltre, abbiamo bisogno di metodi di training adatti a
  - dati non stazionari
  - dati non-iid

# Metodi Incrementali

# Stochastic Gradient Descent in Value Function Approximation

- Goal – trovare il vettore dei parametri ***w*** minimizzando l'errore quadrato medio tra l'approximate value $\hat{v}(s; \mathbf{w})$ e il true value function $v_\pi(s)$

$$\overline{\mathrm{VE}}(\mathbf{w}) \doteq \sum_{s \in \mathcal{S}} \mu(s) \Big[ v_\pi(s) - \hat{v}(s, \mathbf{w}) \Big]^2 \qquad \mu(s) \geq 0,\ \sum_s \mu(s) = 1$$

Quali stati sono più rilevanti

- Gradient solution

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t - \frac{1}{2} \alpha \nabla \Big[ v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t) \Big]^2$$

$$= \mathbf{w}_t + \alpha \Big[ v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t) \Big] \nabla \hat{v}(S_t, \mathbf{w}_t)$$

$$\begin{aligned}
\frac{\partial E}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2n} \sum_{i=1}^{n} (t_i - o_i)^2 \\
&= \frac{1}{2n} \sum_{i=1}^{n} \frac{\partial}{\partial w_j} (t_i - o_i)^2 && \text{[chain rule]} \\
&= \frac{1}{2n} \sum_{i=1}^{n} 2(t_i - o_i) \frac{\partial}{\partial w_j} (t_i - o_i) && \text{[sum rule]} \\
&= \frac{1}{n} \sum_{i=1}^{n} (t_i - o_i) \left( \frac{\partial}{\partial w_j} t_i - \frac{\partial}{\partial w_j} o_i \right) \\
&= -\frac{1}{n} \sum_{i=1}^{n} (t_i - o_i) \frac{\partial}{\partial w_j} o_i .
\end{aligned}$$

$$\boxed{\nabla f(\mathbf{w}) \doteq \left( \frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)^\top}$$

gradiente di *f* rispetto a **w**

# Feature Vector State

- Rappresentare lo stato con un feature vector

$$\boldsymbol{x}(S) = \begin{bmatrix} x_1(S) \\ \vdots \\ x_n(S) \end{bmatrix}$$

- Ad esempio:
  - Neural embedding
  - Distanza dei robot da punti di riferimento
  - Trend del mercato azionario

# Linear Value Function Approximation

- Il valore viene rappresentato come una combinazione lineare delle feature di uno stato

$$\hat{v}(s,\mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s) \doteq \sum_{i=1}^{d} w_i x_i(s)$$

- La funzione obiettivo è quadratica rispetto ai parametri ***w***
  - Stochastic gradient descent converge all'ottimo globale
  - Simple Update Rule (Least Mean Square)

$$\nabla \hat{v}(s,\mathbf{w}) = \mathbf{x}(s)$$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[ U_t - \hat{v}(S_t,\mathbf{w}_t) \Big] \mathbf{x}(S_t)$$

# Lookup Table & Feature

- Un caso speciale di linear value function approximation
- Utilizzo delle table lookup feature

$$x(S) = \begin{bmatrix} \mathbf{1}(S; s_1) \\ \vdots \\ \mathbf{1}(S; s_n) \end{bmatrix}$$

- Vettore dei parametri ***w*** per ogni stato

$$\hat{v}(S; \mathbf{w}) = \begin{bmatrix} \mathbf{1}(S; s_1) \\ \vdots \\ \mathbf{1}(S; s_n) \end{bmatrix}^T \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

# Prediction Incrementale

# Algoritmi di Incremental Prediction

- Finora abbiamo ipotizzato l'accesso al true value $v_\pi(s)$, ma nel RL non c'è alcun supervisore, solo ricompense

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\Big[\underline{v_\pi(S_t)} - \hat{v}(S_t,\mathbf{w}_t)\Big]\nabla\hat{v}(S_t,\mathbf{w}_t)$$

- In pratica, sostituiamo $v_\pi(s)$ con un target
  - MC – Il target è il guadagno $G_t$

$$\Delta \boldsymbol{w} = \alpha\big(G_t - \hat{v}(S_t;\boldsymbol{w})\big)\nabla_{\boldsymbol{w}}\hat{v}(S_t;\boldsymbol{w})$$

  - TD(0) – Il target è il TD target $R_{t+1} + \gamma\hat{v}(S_{t+1};\boldsymbol{w})$

$$\Delta \boldsymbol{w} = \alpha\big(R_{t+1} + \gamma\hat{v}(S_{t+1};\boldsymbol{w}) - \hat{v}(S_t;\boldsymbol{w})\big)\nabla_{\boldsymbol{w}}\hat{v}(S_t;\boldsymbol{w})$$

  - TD(λ) – Il target è λ-return $G_t^\lambda$

$$\Delta \boldsymbol{w} = \alpha\left(G_t^\lambda - \hat{v}(S_t;\boldsymbol{w})\right)\nabla_{\boldsymbol{w}}\hat{v}(S_t;\boldsymbol{w})$$

# Value Function Approximation - MC

- Il guadagno $G_t$ è un campione non distorto e rumoroso del true value $v_\pi(S_t)$
- Si può quindi applicare l'apprendimento supervisionato sui campioni di addestramento

$$\langle S_1, G_1 \rangle, \langle S_2, G_2 \rangle, \dots, \langle S_T, G_T \rangle$$

- Linear Monte-Carlo policy evaluation

$$\Delta \boldsymbol{w} = \alpha\big(G_t - \hat{v}(S_t; \boldsymbol{w})\big)\nabla_{\boldsymbol{w}} \hat{v}(S_t; \boldsymbol{w}) = \alpha\big(G_t - \hat{v}(S_t; \boldsymbol{w})\big)\boldsymbol{x}(S_t)$$

- La Monte-Carlo evaluation converge verso un ottimo locale
- Anche quando si utilizza una non-linear value function approximation

# Value Function Approximation - MC

**Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$**

Input: the policy $\pi$ to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \rightarrow \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using $\pi$
    Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[G_t - \hat{v}(S_t,\mathbf{w})\big]\nabla\hat{v}(S_t,\mathbf{w})$

# Value Function Approximation - TD

- Il TD-target $R_{t+1} + \gamma \hat{v}(S_{t+1}; \boldsymbol{w})$ è un campione distorto del true value $v_\pi(S_t)$
  - Bootstrapping targets
- Si può ancora applicare l'apprendimento supervisionato sui campioni di addestramento

$$\langle S_1, R_2 + \gamma \hat{v}(S_2; \boldsymbol{w}) \rangle, \dots, \langle S_{T-1}, R_T \rangle$$

- Non producono un vero metodo gradient-descent (sono chiamati metodi *semi-gradient*)
- Linear TD(0) policy evaluation

$$\Delta \boldsymbol{w} = \alpha\big(R + \gamma \hat{v}(S'; \boldsymbol{w}) - \hat{v}(S; \boldsymbol{w})\big)\nabla_{\boldsymbol{w}} \hat{v}(S; \boldsymbol{w}) = \alpha \delta \boldsymbol{x}(S)$$

- Linear TD(0) converge (vicino) all'ottimo globale

# Value Function Approximation - TD

**Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$**

Input: the policy $\pi$ to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \rightarrow \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A \sim \pi(\cdot|S)$
        Take action $A$, observe $R, S'$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
        $S \leftarrow S'$
    until $S$ is terminal

# Value Function Approximation – TD(λ)

- λ-return $G_t^\lambda$ è un campione distorto del true value $v_\pi(S_t)$
- Si può ancora applicare l'apprendimento supervisionato sui campioni di addestramento

$$\langle S_1, G_1^\lambda \rangle, \dots, \langle S_{T-1}, G_{T-1}^\lambda \rangle$$

- Forward view linear TD(λ)

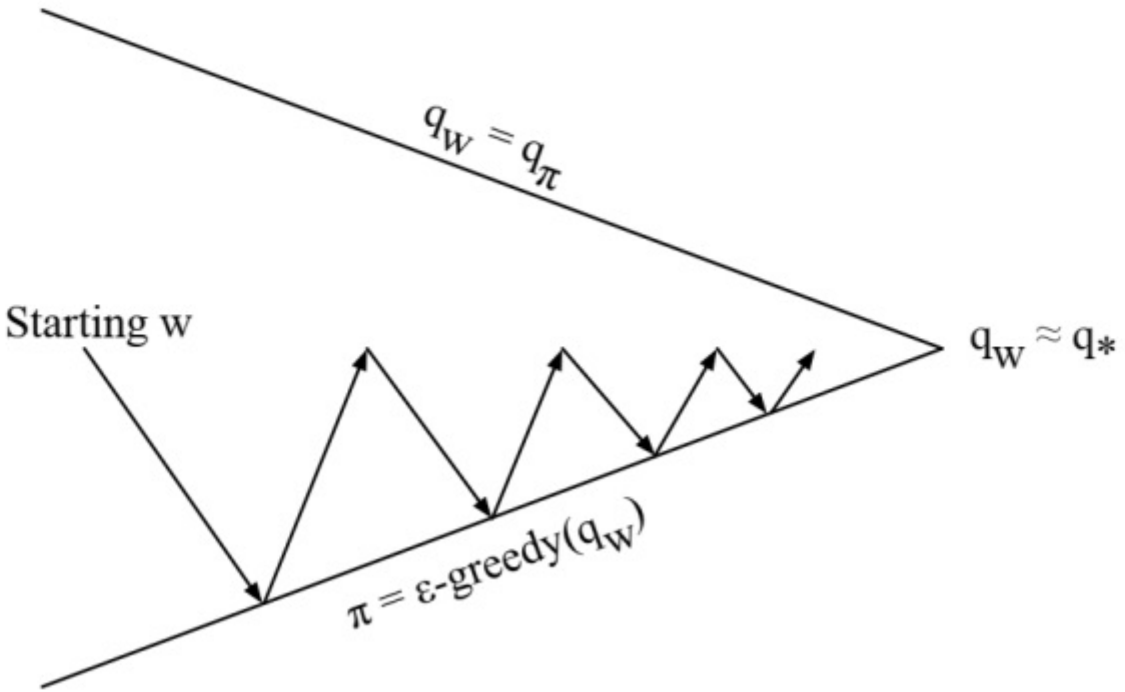
$$\Delta \boldsymbol{w} = \alpha \left( G_t^\lambda - \hat{v}(S_t; \boldsymbol{w}) \right) \nabla_{\boldsymbol{w}} \hat{v}(S_t; \boldsymbol{w}) = \alpha \left( G_t^\lambda - \hat{v}(S_t; \boldsymbol{w}) \right) \boldsymbol{x}(S_t)$$

- Backward view linear TD(λ)

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}; \boldsymbol{w}) - \hat{v}(S_t; \boldsymbol{w})$$
$$E_t = \lambda \gamma E_{t-1} + \boldsymbol{x}(S_t)$$
$$\Delta \boldsymbol{w} = \alpha \delta_t E_t$$

# Control Incrementale

# Control con Value Function Approximation




- Valutazione della Policy – Approximate policy evaluation, $\hat{q}(\cdot, \cdot; \boldsymbol{w}) \approx q_\pi(\cdot,\cdot)$
- Miglioramento della Policy – $\epsilon$-greedy policy improvement

# Action-Value Function Approximation

- Approssimare la action-value function $\hat{q}(S, A; \boldsymbol{w}) \approx q_\pi(S, A)$
- Minimizzare il MSE tra il approximate action-value $\hat{q}(S, A; \boldsymbol{w})$ e il true action-value $q_\pi(S, A)$

$$J(\boldsymbol{w}) = \mathbb{E}_\pi\left[\left(q_\pi(S, A) - \hat{q}(S, A; \boldsymbol{w})\right)^2\right]$$

- Utilizzare stochastic gradient descent per trovare un minimo locale

$$-\frac{1}{2}\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \left(q_\pi(S, A) - \hat{q}(S, A; \boldsymbol{w})\right)\nabla_{\boldsymbol{w}}\hat{q}(S, A; \boldsymbol{w})$$

$$\Delta \boldsymbol{w} = \alpha \left(q_\pi(S, A) - \hat{q}(S, A; \boldsymbol{w})\right)\nabla_{\boldsymbol{w}}\hat{q}(S, A; \boldsymbol{w})$$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\left[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\right]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t)$$

# Linear Action-Value Function Approximation

- Utilizzo delle table feature action-states

$$x(S,A) = \begin{bmatrix} x_1(S,A) \\ \vdots \\ x_n(S,A) \end{bmatrix}$$

- Rappresenta la action-value function tramite una combinazione lineare delle feature

$$\hat{q}(S,A;\mathbf{w}) = \mathbf{x}(S,A)^T \mathbf{w}$$

- Stochastic gradient descent update

$$\nabla_{\mathbf{w}} \hat{q}(S,A;\mathbf{w}) = \mathbf{x}(S,A)$$
$$\Delta \mathbf{w} = \alpha \left( q_\pi(S,A) - \hat{q}(S,A;\mathbf{w}) \right) \mathbf{x}(S,A)$$

# Algoritmi di Incremental Control

**Anche in questo caso abbiamo bisogno di un non-oracular target per $q_\pi(S, A)$**

- MC – Il target è il guadagno $G_t$

$$\Delta \boldsymbol{w} = \alpha\left(G_t - \hat{q}(S_t, A_t; \boldsymbol{w})\right)\nabla_{\boldsymbol{w}}\hat{q}(S_t, A_t; \boldsymbol{w})$$

- TD(0) – Il target è il TD target $R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}; \boldsymbol{w})$

$$\Delta \boldsymbol{w} = \alpha\left(R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}; \boldsymbol{w}) - \hat{q}(S_t, A_t; \boldsymbol{w})\right)\nabla_{\boldsymbol{w}}\hat{q}(S_t, A_t; \boldsymbol{w})$$

- Forward TD(λ) – Il target è l'action-value λ-return

$$\Delta \boldsymbol{w} = \alpha\left(q_t^\lambda - \hat{q}(S_t, A_t; \boldsymbol{w})\right)\nabla_{\boldsymbol{w}}\hat{q}(S_t, A_t; \boldsymbol{w})$$

- Backward TD(λ) – Il target rimane invariato

$$\delta_t = R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}; \boldsymbol{w}) - \hat{q}(S_t, A_t; \boldsymbol{w})$$
$$E_t = \lambda\gamma E_{t-1} + \nabla_{\boldsymbol{w}}\hat{q}(S_t, A_t; \boldsymbol{w})$$
$$\Delta \boldsymbol{w} = \alpha\delta_t E_t$$

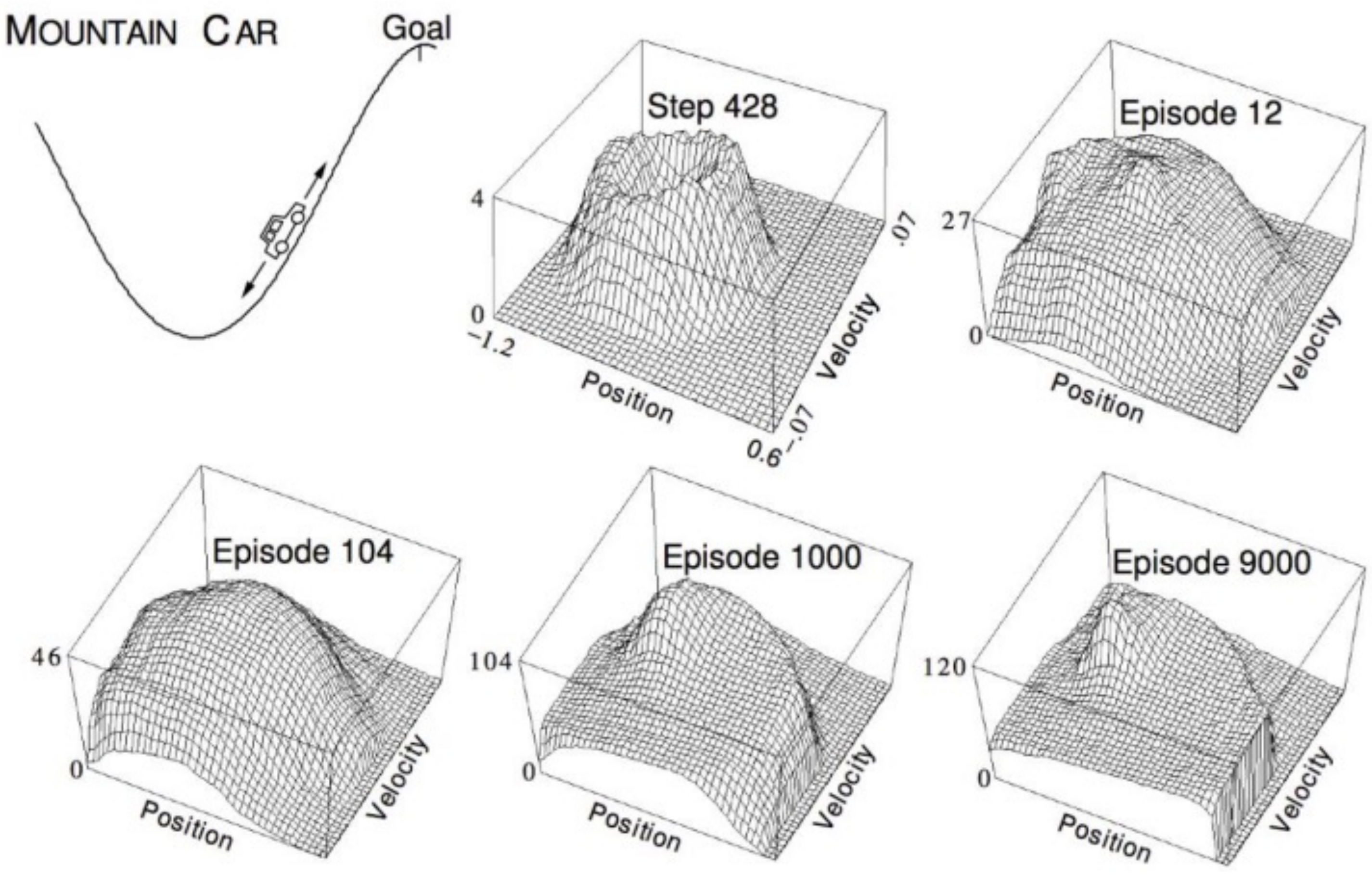
# Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q^*$

**Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$**

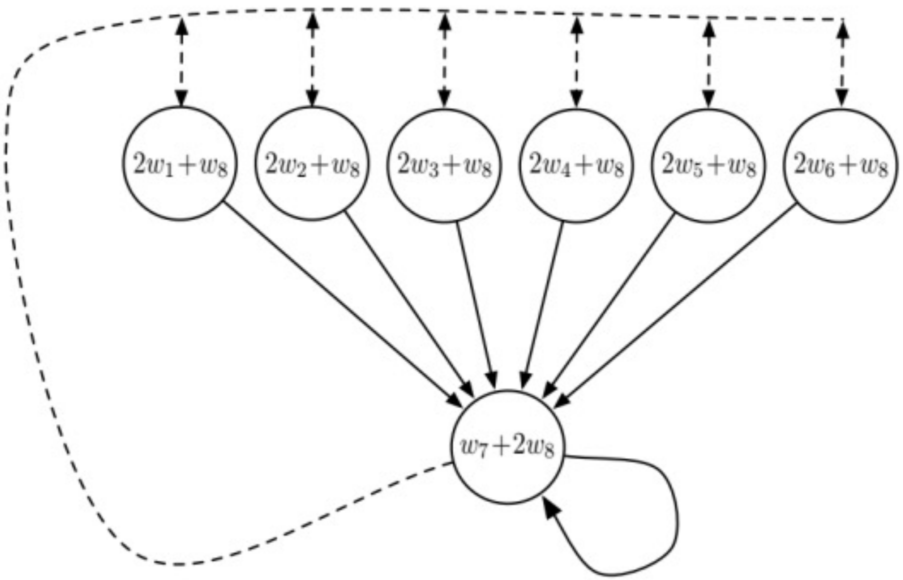
Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    $S, A \leftarrow$ initial state and action of episode (e.g., $\varepsilon$-greedy)
    Loop for each step of episode:
        Take action $A$, observe $R, S'$
        If $S'$ is terminal:
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
            Go to next episode
        Choose $A'$ as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., $\varepsilon$-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

# Linear SARSA with Coarse Coding in Mountain Car



# Baird's Counterexample

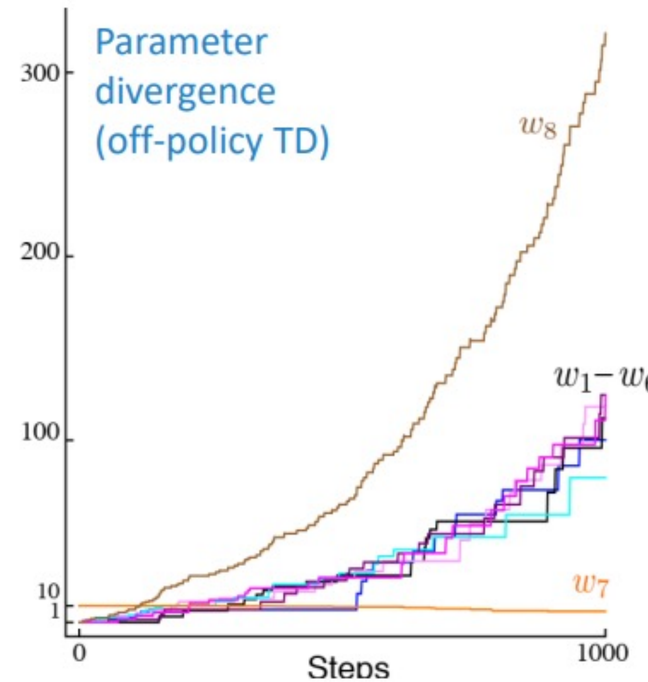



$\pi(\text{solid}|\cdot) = 1$

$b(\text{dashed}|\cdot) = 6/7$

$b(\text{solid}|\cdot) = 1/7$

$\gamma = 0.99$



# La triade letale

- Function approximation
- Bootstrapping
- Off-policy training

# Convergenza degli Algoritmi di Prediction

| On/Off-Policy | Algorithm | Table Lookup | Linear | Non-Linear |
|---|---|---|---|---|
| On-Policy | MC | ✓ | ✓ | ✓ |
| | TD(0) | ✓ | ✓ | ✗ |
| | TD($\lambda$) | ✓ | ✓ | ✗ |
| Off-Policy | MC | ✓ | ✓ | ✓ |
| | TD(0) | ✓ | ✗ | ✗ |
| | TD($\lambda$) | ✓ | ✗ | ✗ |