UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Intelligenza Artificiale

Value Function Approximation

a.a. 2022/2023

Metodi Batch

Batch Reinforcement Learning

- Gradient descent semplice ed accattivante
- Ma non è efficiente dal punto di vista del campionamento
- I metodi batch cercano di trovare la value function più adatta
- Data l'esperienza dell'agente (dati di addestramento)

Least Squares Prediction

- Data una value function approximation $\hat{v}(s; \boldsymbol{w}) \approx v_\pi(s)$ e l'esperienza *D* costituita da coppie <*stato*, *valore*>

$$\mathcal{D} = \{\langle S_1, v_1^\pi \rangle, \langle S_2, v_2^\pi \rangle, \dots, \langle S_T, v_T^\pi \rangle\}$$

- Quali parametri $\boldsymbol{w}$ offrono il miglior risultato di adattamento della value function $\hat{v}(s; \boldsymbol{w})$?
- Gli algoritmi Least Squares individuano il vettore dei parametri $\boldsymbol{w}$ che minimizza l'errore quadratico medio tra $\hat{v}(s; \boldsymbol{w})$ e i valori target $v_\pi(s)$

$$LS(\boldsymbol{w}) = \sum_{t=1}^{T} \left(v^\pi - \hat{v}(s_t; \boldsymbol{w})\right)^2 = \mathbb{E}_{\mathcal{D}} \left[\left(v^\pi - \hat{v}(s; \boldsymbol{w})\right)^2\right]$$

Experience Replay

SGD con Experience Replay

- Data una value function approximation $\hat{v}(s; \boldsymbol{w}) \approx v_\pi(s)$ e l'esperienza $\mathcal{D}$ costituita da coppie <*stato*, *valore*>

$$\mathcal{D} = \{\langle S_1, v_1^\pi \rangle, \langle S_2, v_2^\pi \rangle, \dots, \langle S_T, v_T^\pi \rangle\}$$

- Ripetere
  1. Campiona stato e valore dall'esperienza *D*

$$\langle s, v^\pi \rangle \sim \mathcal{D}$$

  2. Esegui lo stochastic gradient descent update

$$\Delta \boldsymbol{w} = \alpha(v^\pi - \hat{v}(s; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{v}(s; \boldsymbol{w})$$

- Converge alla soluzione least squares

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}} LS(\boldsymbol{w})$$

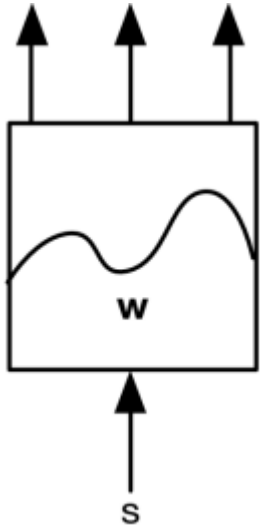
Deep Q-Networks (DQN)

- DQN usa experience replay e Q-target fissati
- Esegue l'azione a_t in base alla policy ϵ-greedy
- Memorizza le transizioni $(s_t, a_t, r_{t+1}, s_{t+1})$ nella replay memory *D*
- Campiona un mini-batch casuale di transizioni (*s*, *a*, *r*, *s'*)
- Calcola i Q-learning target rispetto ai vecchi parametri fissati w^-
- Ottimizza la MSE tra la Q-network e i Q-learning target

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

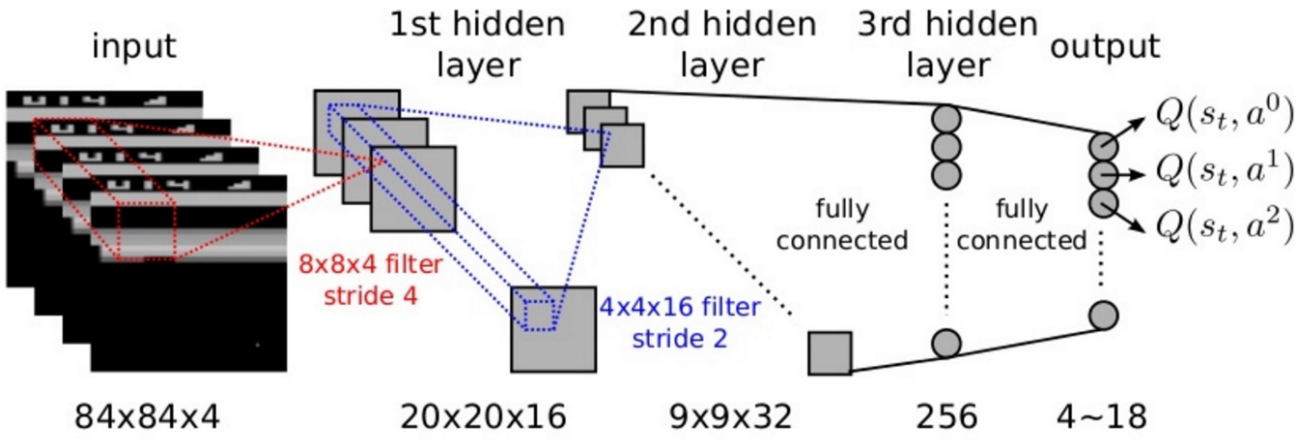
- Utilizza una variante del *stochastic gradient descent*

Atari-DQN

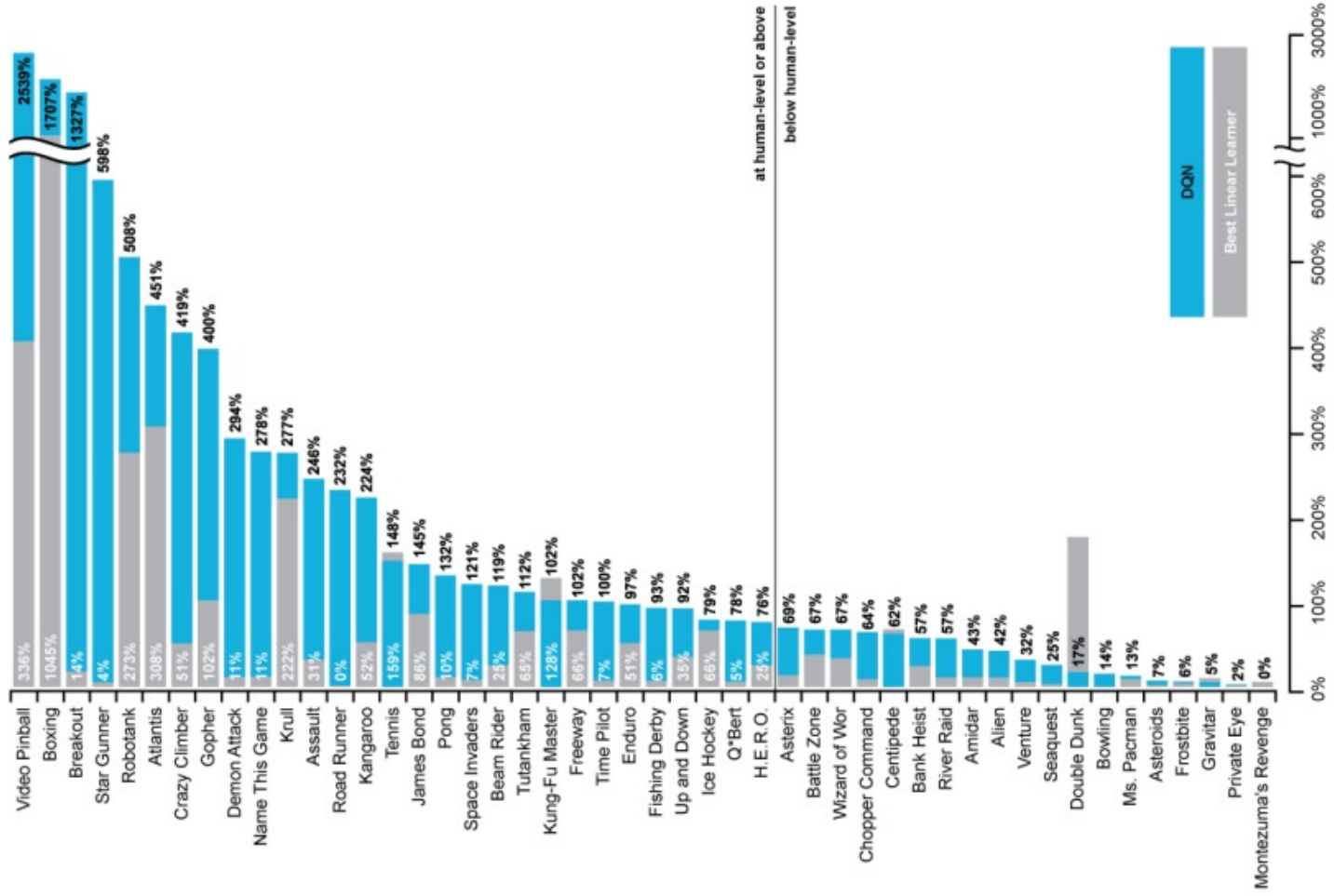


- Apprendimento end-to-end dei valori $Q(s, a)$ dai pixel s
- Lo stato di input s è uno stack di pixel grezzi degli ultimi 4 fotogrammi
- L'output è $Q(s, a)$ per 18 posizioni di joystick/pulsante
- La ricompensa è la variazione del punteggio per quello step

L'Architettura di rete e gli iperparametri sono gli stessi per tutti i giochi



Atari-DQN (Risultati)



Least Squares Prediction and Control

Linear Least Squares Prediction

- L'experience replay trova una soluzione least squares, ma può richiedere molte iterazioni
- Utilizzo della linear value function approximation $\hat{v}(s; \boldsymbol{w}) = \mathbf{x}(s)^T \mathbf{w}$
- Possiamo risolvere la soluzione least squares direttamente

Linear Least Squares Prediction - Batch

- Al valore minimo di $LS(\boldsymbol{w})$, l'expected update deve essere pari a zero

$$\mathbb{E}_{\mathcal{D}}[\Delta \boldsymbol{w}] = 0$$

$$\alpha \sum_{t=1}^{T} \boldsymbol{x}(S_t)(v_t^{\pi} - \boldsymbol{x}(S_t)^T \boldsymbol{w}) = 0$$

$$\boldsymbol{w} = \left(\sum_{t=1}^{T} \boldsymbol{x}(S_t)\boldsymbol{x}(S_t)^T \right)^{-1} \sum_{t=1}^{T} \boldsymbol{x}(S_t) v_t^{\pi}$$

- Costi diretti (N^3) e incrementali (N^2)

Linear Least Squares Prediction - Algoritmi

- Non conosciamo il true value v_t^π
- I dati di addestramento utilizzano campioni rumorosi o distorti di v_t^π
  - Least Squares Monte-Carlo (LSMC) usa return $v_t^\pi \approx G_t$
  - Least Squares TD (LSTD) usa TD target $v_t^\pi \approx R_{t+1} + \gamma\hat{v}(S_{t+1}; \boldsymbol{w})$
  - Least Squares TD(λ) (LSTD(λ)) usa λ-return $v_t^\pi \approx G_t^\lambda$
- In ogni caso bisogna risolvere direttamente il punto fisso di MC/TD/TD(λ)

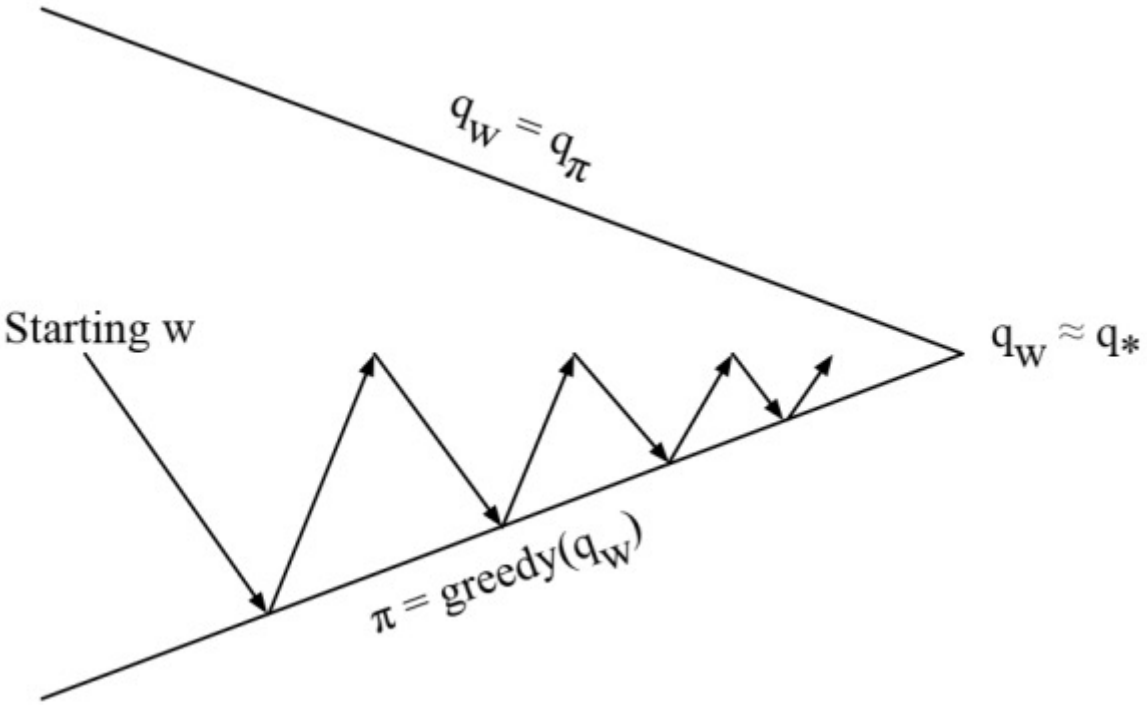
LS Updates

LSMC	$\boldsymbol{w} = \left(\sum_{t=1}^{T} \boldsymbol{x}(S_t) \boldsymbol{x}(S_t)^T \right)^{-1} \sum_{t=1}^{T} \boldsymbol{x}(S_t) G_t$
LSTD	$\boldsymbol{w} = \left(\sum_{t=1}^{T} \boldsymbol{x}(S_t) \left(\boldsymbol{x}(S_t) - \gamma \boldsymbol{x}(S_{t+1}) \right)^T \right)^{-1} \sum_{t=1}^{T} \boldsymbol{x}(S_t) R_{t+1}$
LSTD(λ)	$\boldsymbol{w} = \left(\sum_{t=1}^{T} E_t \left(\boldsymbol{x}(S_t) - \gamma \boldsymbol{x}(S_{t+1}) \right)^T \right)^{-1} \sum_{t=1}^{T} \boldsymbol{x}(S_t) R_{t+1}$

Convergenza degli Algoritmi di Linear Least Squares Prediction

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	✗
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✗	✗
	LSTD	✓	✓	-

Least Squares Control



- Valutazione della Policy – Policy evaluation attraverso least squares Q-learning
- Miglioramento della Policy – Greedy policy improvement

Least Squares Action-Value Function Approximation

- Approssimare $q_\pi(s, a)$ usando combinazioni lineari delle feature $\boldsymbol{x}(s, a)$

$$\hat{q}(s, a; \boldsymbol{w}) = \boldsymbol{x}(s, a)^T \boldsymbol{w} \approx q_\pi(s, a)$$

- Minimizzare il least squares error tra $\hat{q}(s, a; \boldsymbol{w})$ e $q_\pi(s, a)$
  - Dall'esperienza generata usando la policy π
  - Composta da coppie <(*stato*, *azione*), *valore*>

$$\mathcal{D} = \{\langle (s_1, a_1), v_1^\pi \rangle, \dots, \langle (s_T, a_T), v_T^\pi \rangle\}$$

Least Squares Control

- Per la policy evaluation, vogliamo utilizzare in modo efficiente tutta l'esperienza acquisita
- Per il control, vogliamo anche migliorare la policy
- Questa esperienza è generata da molte policy
- Quindi, per valutare $q_\pi(s, a)$ dobbiamo apprendere off-policy
- Utilizziamo la stessa idea del Q-learning:
  - Utilizziamo l'esperienza generata dalla vecchia policy $S_t, A_t, R_{t+1}, S_{t+1} \sim \pi_{old}$
  - Consideriamo un'azione alternativa $A' \sim \pi_{new}(S_{t+1})$
  - Aggiorniamo $\hat{q}(S_t, A_t; \boldsymbol{w})$ rispetto al valore dell'azione alternativa $R_{t+1} + \gamma \hat{q}(S_{t+1}, A'; \boldsymbol{w})$

Least Squares Q-Learning

- Si consideri il seguente linear Q-learning update

$$\delta_t = R_{t+1} + \gamma \hat{q}(S_{t+1}, \pi(S_{t+1}); \boldsymbol{w}) - \hat{q}(S_t, A_t; \boldsymbol{w})$$
$$\Delta \boldsymbol{w}_t = \alpha \delta_t \boldsymbol{x}(S_t, A_t)$$

- Algoritmo LSTDQ: risoluzione per aggiornamento totale uguale a zero

$$\Delta \boldsymbol{w}_t = 0$$
$$\boldsymbol{w} = \left(\sum_{t=1}^{T} \boldsymbol{x}(S_t, A_t) \left(\boldsymbol{x}(S_t, A_t) - \gamma \boldsymbol{x}(S_{t+1}, \pi(S_{t+1})) \right)^T \right)^{-1} \sum_{t=1}^{T} \boldsymbol{x}(S_t, A_t) R_{t+1}$$

Convergenza degli Algoritmi di Control

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
LSPI	✓	(✓)	-

(✓) = si aggira intorno a una value-function quasi ottimale

Take home messages

- Value function approximation: scalare RL a problemi di dimensioni reali
- Utilizzare MC, TD e TD(λ) per generare campioni per l'addestramento degli (action) value approximator
  - Stochastic Gradient Descent (incrementale)
  - Least Squares (batch)
- Experience replay dei dati memorizzati e aggiornamenti con target dei parametri meno recenti (DQN)
- Linear least squares fornisce una soluzione in forma chiusa
- La convergenza dell'apprendimento potrebbe essere complicata