(Deep) Q learning

Jorge Vasquez

¿Cómo encontrar Políticas Óptimas?

- Ecuaciones de *Bellman* para Funciones de Valor
- Métodos:
  - Programación Dinámica
    - Iteración de Valor
    - Iteración de Política
    - Otras versiones
  - Algoritmos
    - Q-Learning
    - Sarsa
    - TD-Learning

Recap

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} \ldots + r_{t+n} + \cdots$$

Recap

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} \ldots + \gamma^{t+n} r_{t+n} + \cdots$$

γ: discount factor; $0 < \gamma < 1$

Q learning

Q-values

- Q*(s,a) = recompensa esperada comenzando en estado s, tomando acción a, y luego, actuar en forma optima.

Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

Q-values

Q-Value Iteration:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$$

Q-Learning

- **Q-value iteration:** $Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$
- **Rewrite as expectation:** $Q_{k+1} \leftarrow \mathbb{E}_{s' \sim P(s'|s,a)} \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$

Q values tabulares

(Tabular) Q-Learning: replace expectation by samples

- For an state-action pair (s,a), receive: $s' \sim P(s'|s,a)$
- Consider your old estimate: $Q_k(s,a)$
- Consider your new sample estimate:

$$\text{target}(s') = R(s,a,s') + \gamma \max_{a'} Q_k(s',a')$$

Q Learning

$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha\left[\text{target}(s')\right]$$

Aprendizaje Q (Q-Learning)

Algorithm:

Start with $Q_0(s,a)$ for all s, a.

Get initial state s

For k = 1, 2, … till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s,a,s')$$

Sample new initial state s'

else:

$$\text{target} = R(s,a,s') + \gamma \max_{a'} Q_k(s',a')$$

$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha\,[\text{target}]$$

¿Como tomamos muestras de las acciones?

- Choose random actions?
- Choose action that maximizes $Q_k(s, a)$ (i.e. greedily)?
- ε-Greedy: choose random action with prob. ε, otherwise choose action greedily

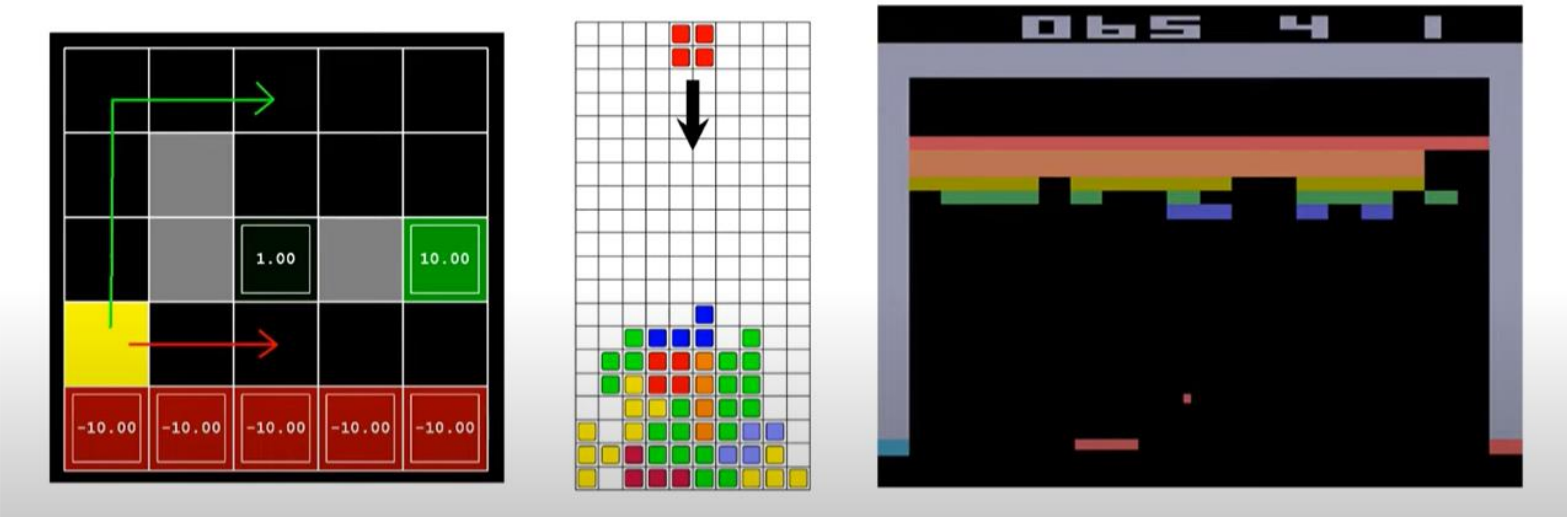
Propiedades de Aprendizaje Q

- Q Learning converge a una política optimal siempre!
- Aprendizaje libre de política

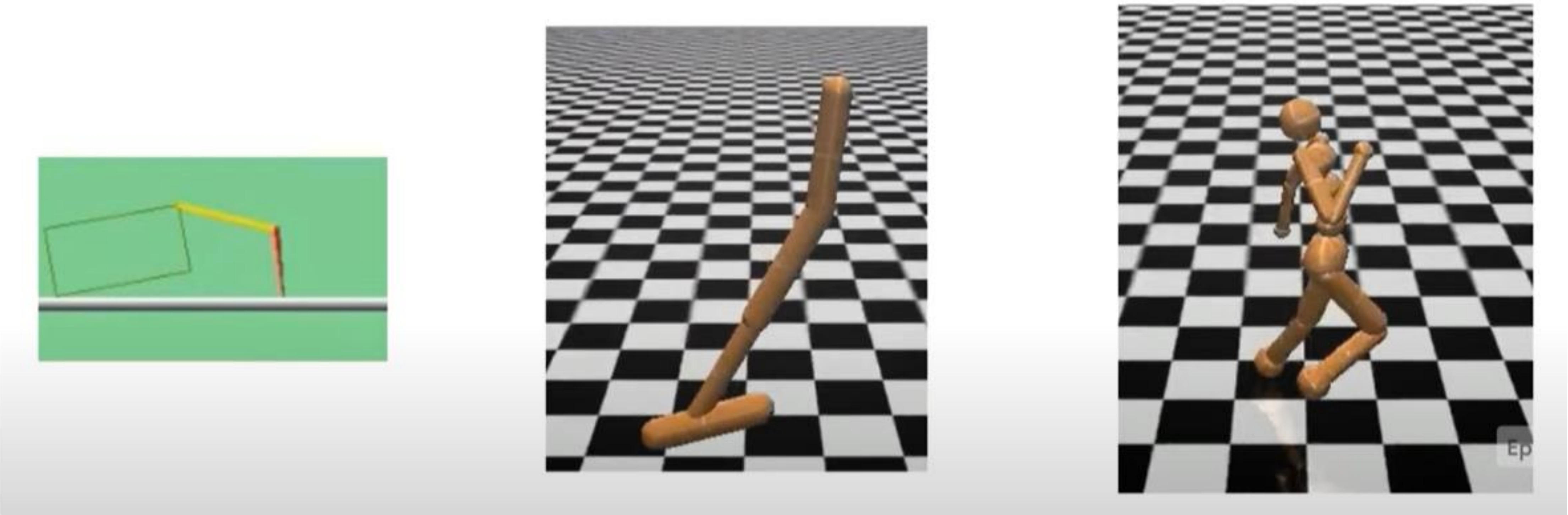
Desventajas del Aprendizaje Q

- Debes explorar suficiente
- Debes configurar la tasa de aprendizaje
  - Hacerlo pequeño, peor no tan luego

Problema en entornos discretos



Problema en entornos continuos



Q-learning Aproximado

$$Q_\theta(s, a)$$

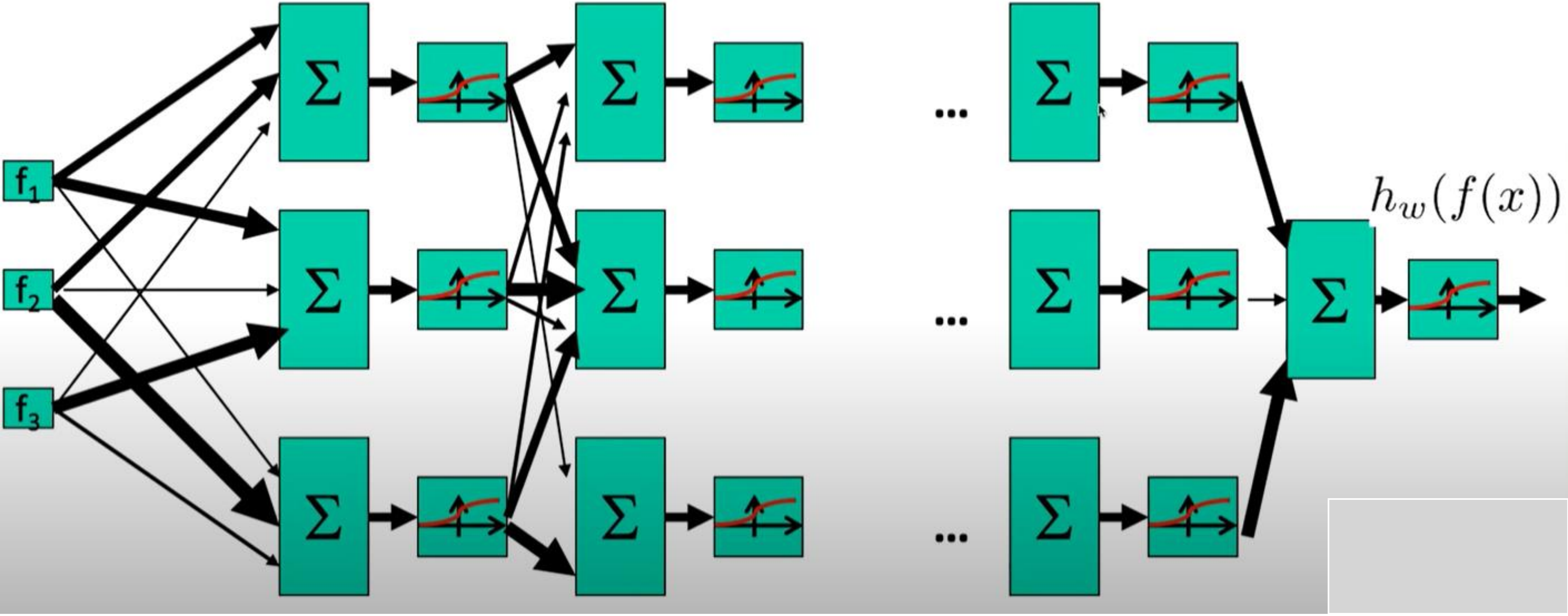
$$Q_\theta(s, a) = \theta_0 f_0(s, a) + \theta_1 f_1(s, a) + \cdots + \theta_n f_n(s, a)$$

$$\text{target}(s') = R(s, a, s') + \gamma \max_{a'} Q_{\theta_k}(s', a')$$

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \left[\frac{1}{2} (Q_\theta(s, a) - \text{target}(s'))^2 \right] \Bigg|_{\theta = \theta_k}$$

Recap de NN

Recap de NN



MLP

$$f(x) = Wx$$

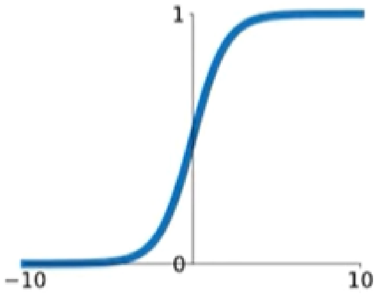
$$f(x) = W_2 \max(0, W_0 x)$$

$$f(x) = W_3 \max(0, W_2 \max(0, W_0 x))$$

MLP

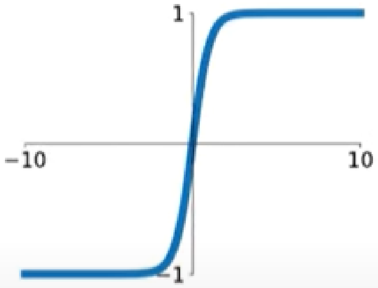
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



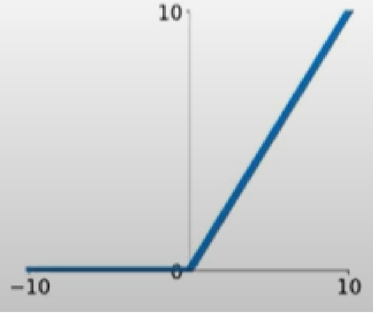
tanh

$\tanh(x)$



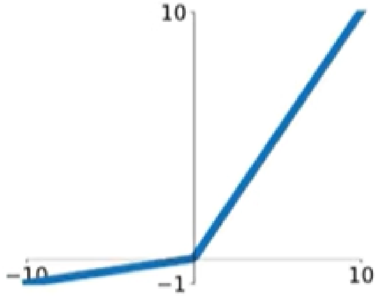
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

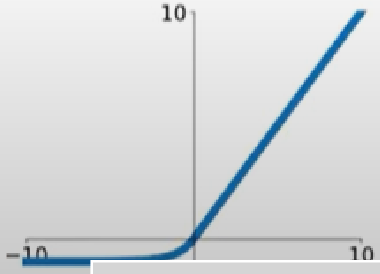


Maxout

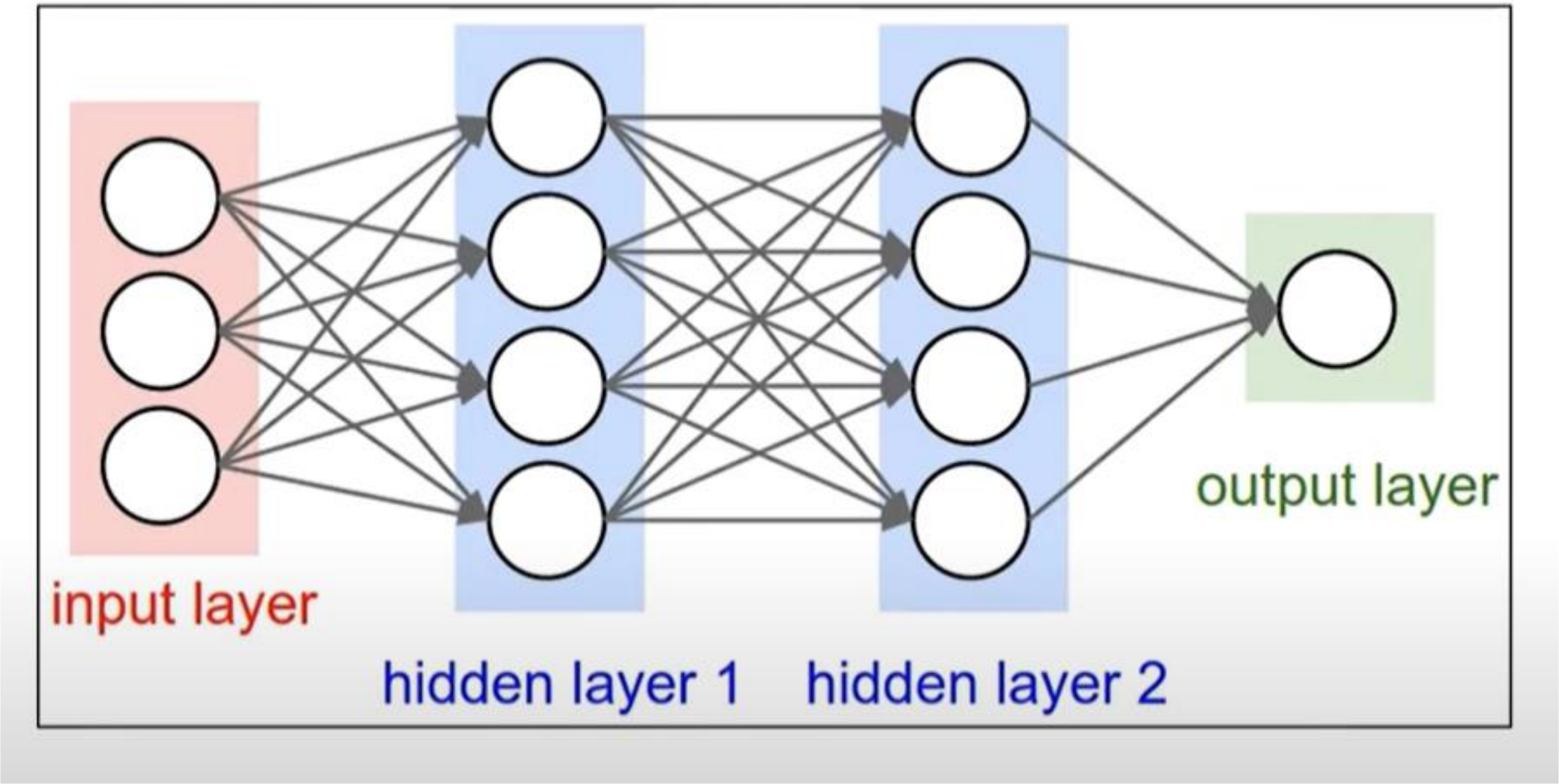
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

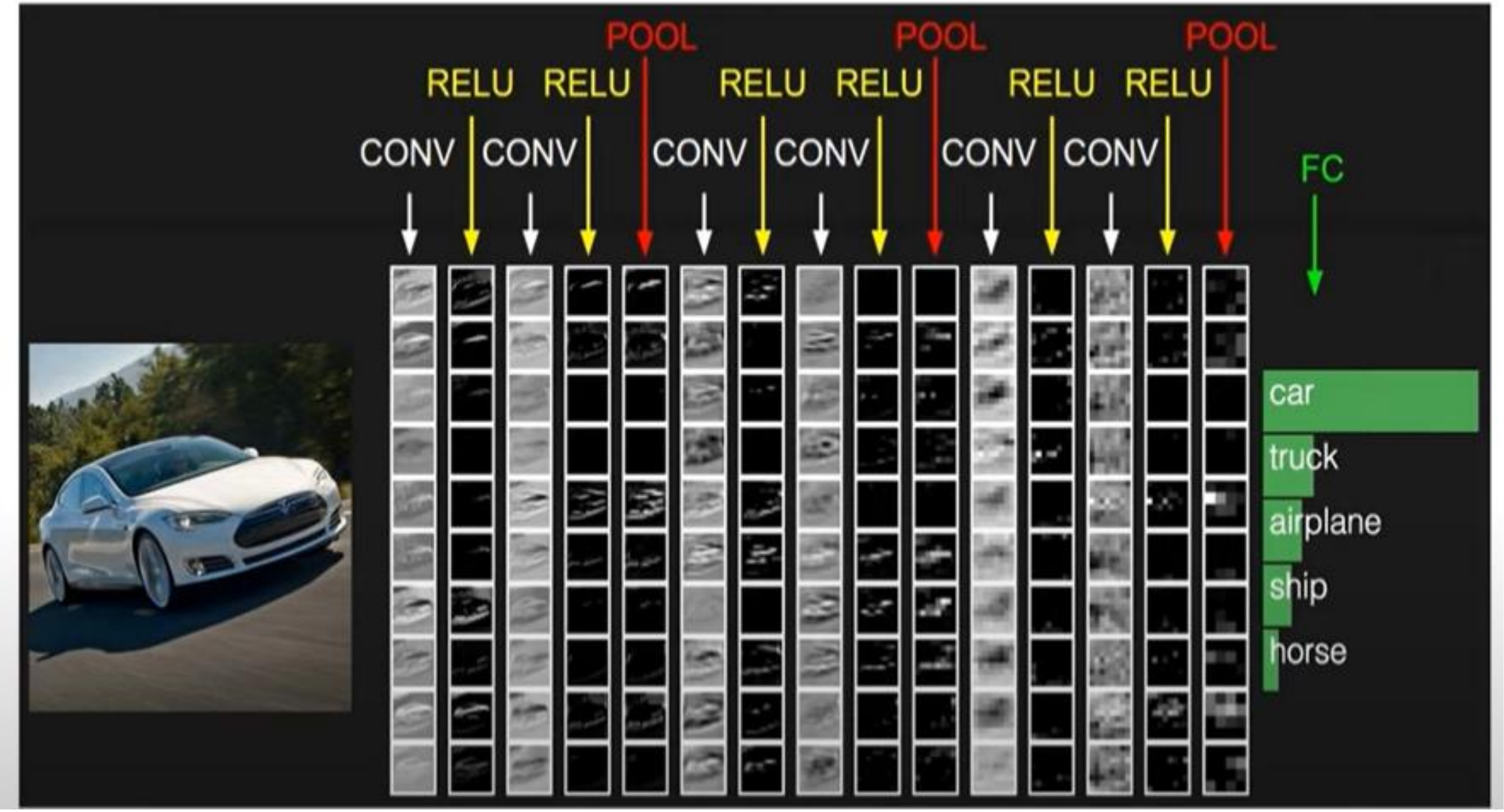
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



MLP



CNN


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Deep Q-Network

$$Q_\theta(s, a)$$

$$\text{target}(s') = R(s, a, s') + \gamma \max_{a'} Q_{\theta_k}(s', a')$$

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \left[\frac{1}{2} (Q_\theta(s, a) - \text{target}(s'))^2 \right]$$

Deep Q-Network

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For t = 1,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

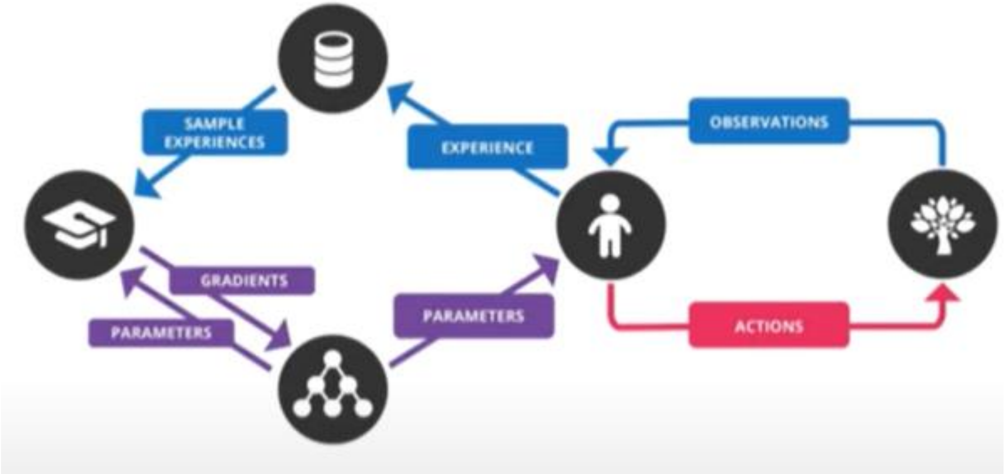
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For



Definición de la Función-Q

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

- La recompensa total Rt es la suma descontada de todas las recompensas obtenidas desde el tiempo t.

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

- La función Q captura el total de recompensas esperadas totales que un agente en estado s puede recibir ejecutando acción a

¿Como tomar acciones dada la función Q?

$$Q(\underbrace{s_t}_{\text{state}}, \underbrace{a_t}_{\text{action}}) = \mathbb{E}[R_t | s_t, a_t]$$

- El agente necesita una política pi, para inferir la mejor acción por tomar en su estado s.
- **Estrategia: La política debe escoger una acción que maximice las recompensas futuras**

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Algoritmos DRL

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Funciones de Valor

- El **Valor de un estado** es la recompensa esperada, comenzando desde es estado. Depende de la política del agente:

State - value function for policy π :

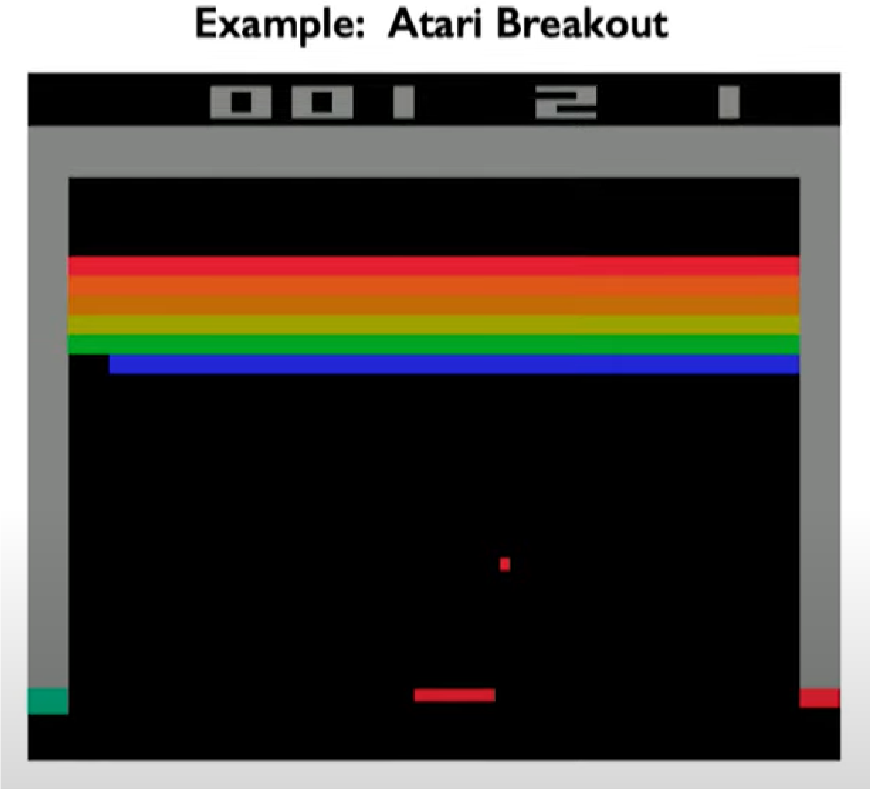
$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

- El valor de tomar una acción en un estado bajo una política π es la recompensa esperada comenzando desde ese estado, tomando esa acction y siguiendo la política π.

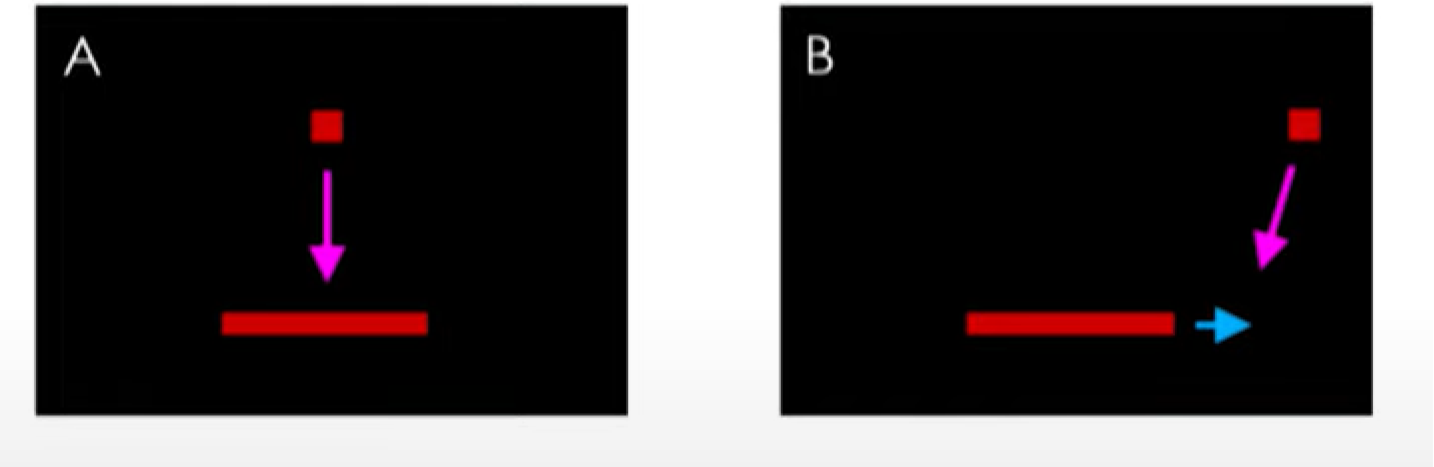
Action - value function for policy π :

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

Función Q

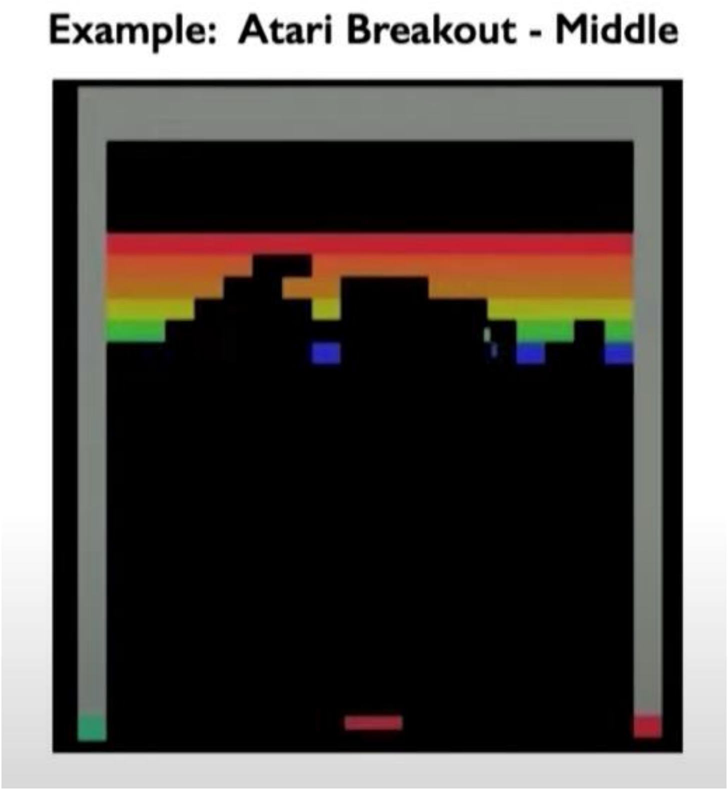


Puede ser muy dificl para los humanos estimar valores Q en forma correcta.

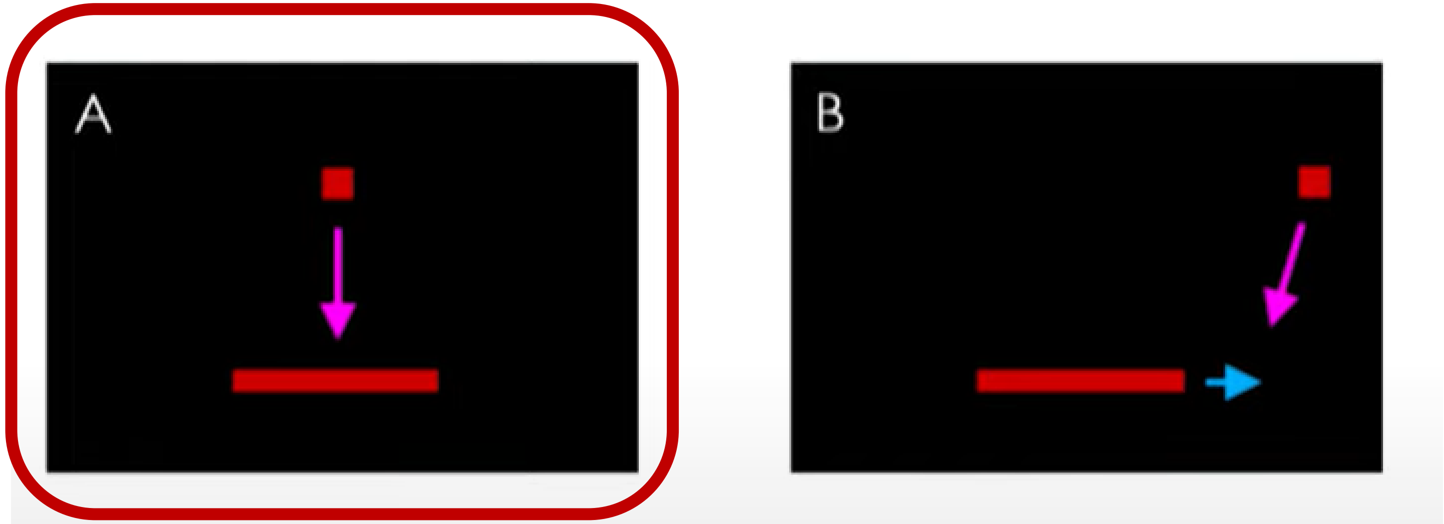


¿Cual par (s,a) tiene un valor-Q mas alto?

Función Q

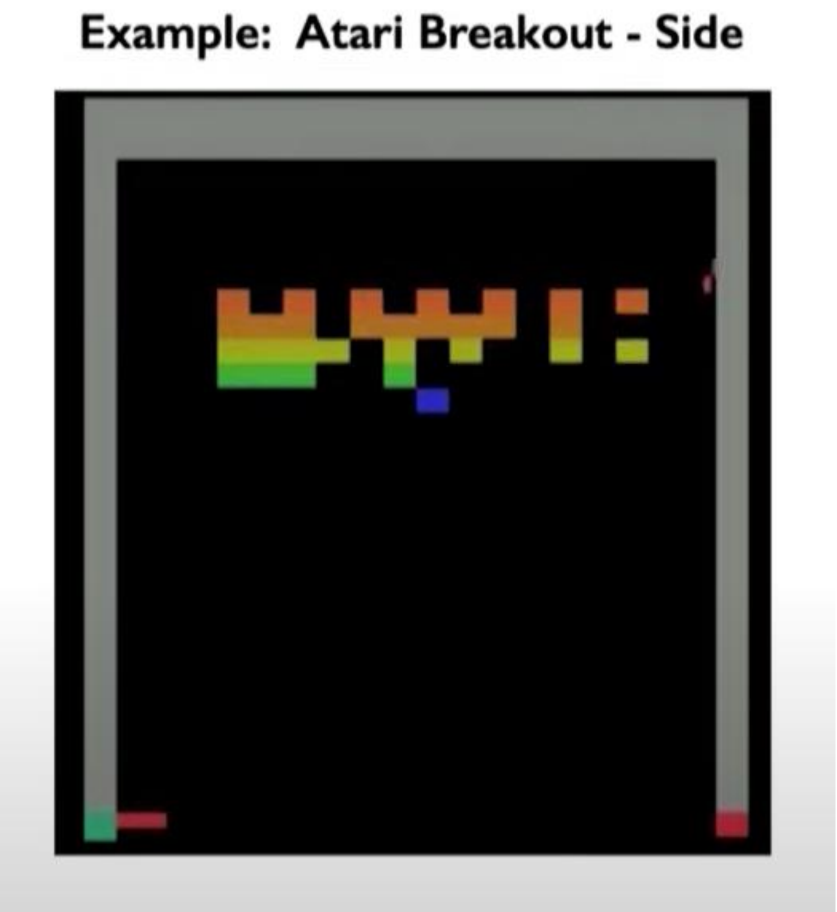


Puede ser muy dificl para los humanos estimar valores Q en forma correcta.

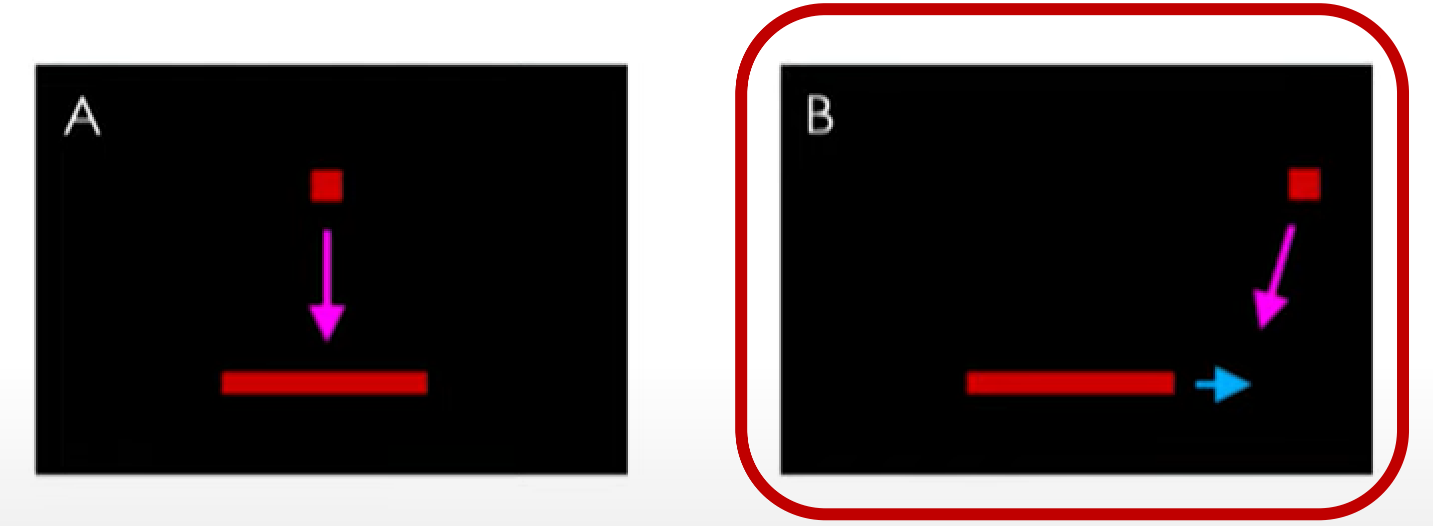


¿Cual par (s,a) tiene un valor-Q mas alto?

Función Q

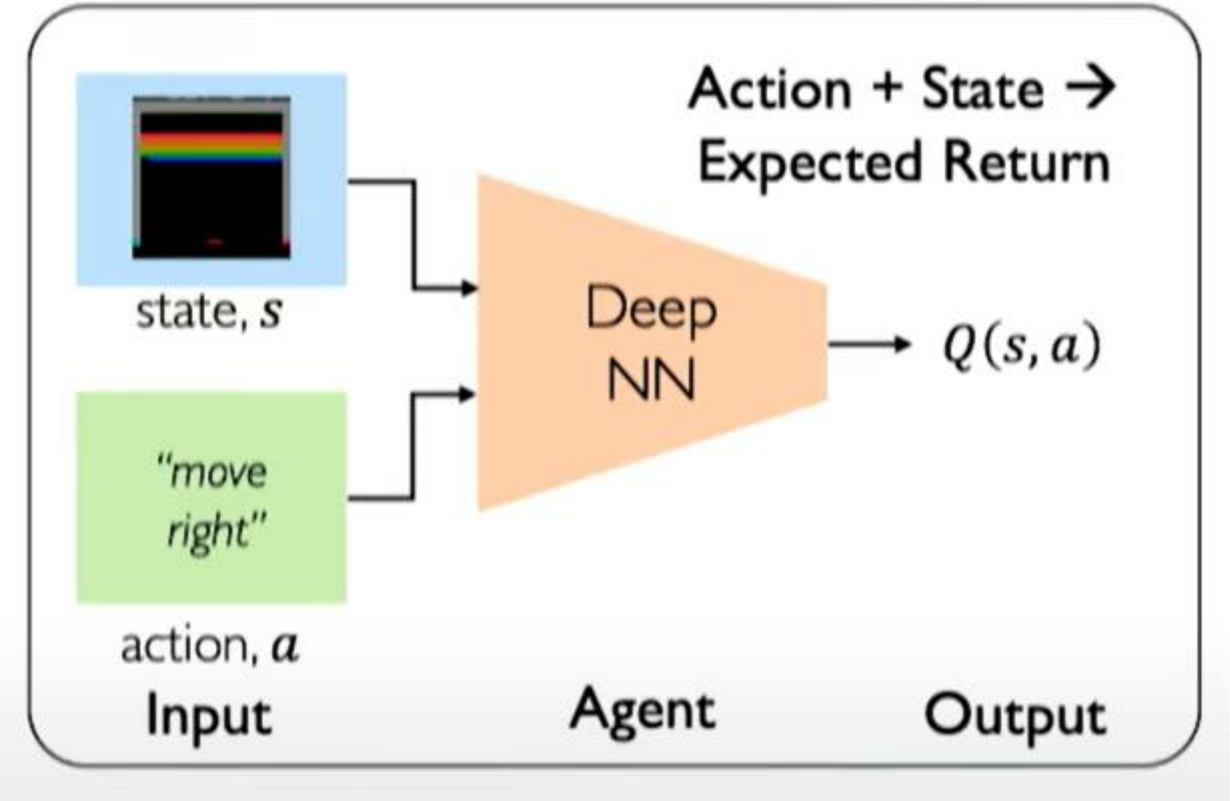


Puede ser muy difícil para los humanos estimar valores Q en forma correcta.



¿Cual par (s,a) tiene un valor-Q mas alto?

Deep Q Networks (DQN)



Deep Q Networks (DQN)