PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

INTELIGENCIA ARTIFICIAL (INF371)

APRENDIZAJE POR REFUERZO (DQN, DDQN)

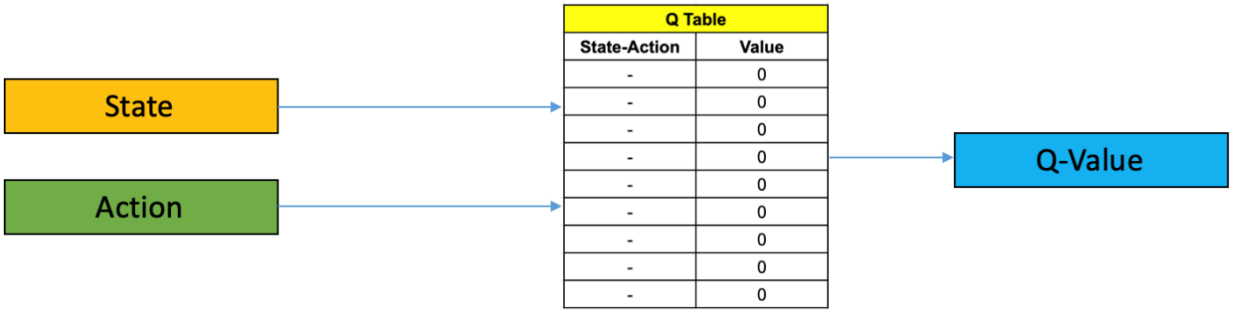
Dr. Edwin Villanueva Talavera

Contenido

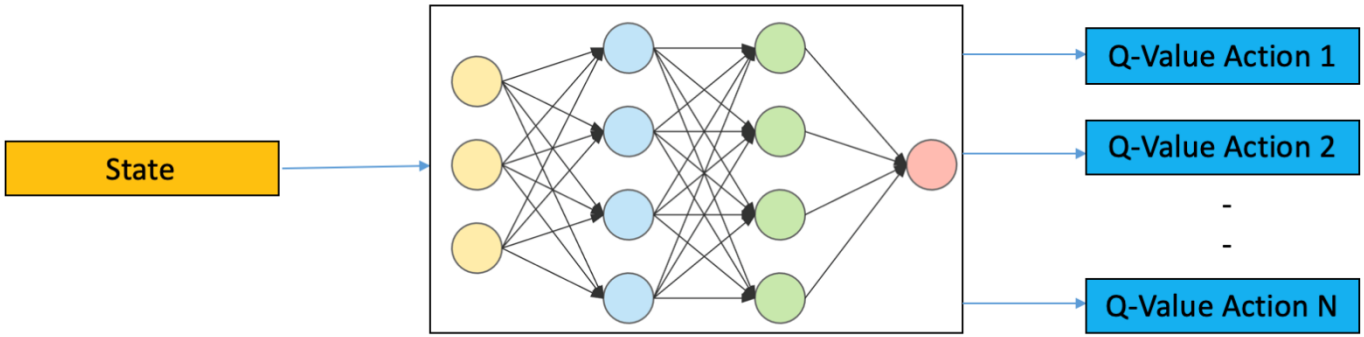
- Deep Q-Networks
- Doble Deep Q-netwotks

Deep Q-networks

- En Deep Q-network se usa una Red neuronal para aproximar los valores Q



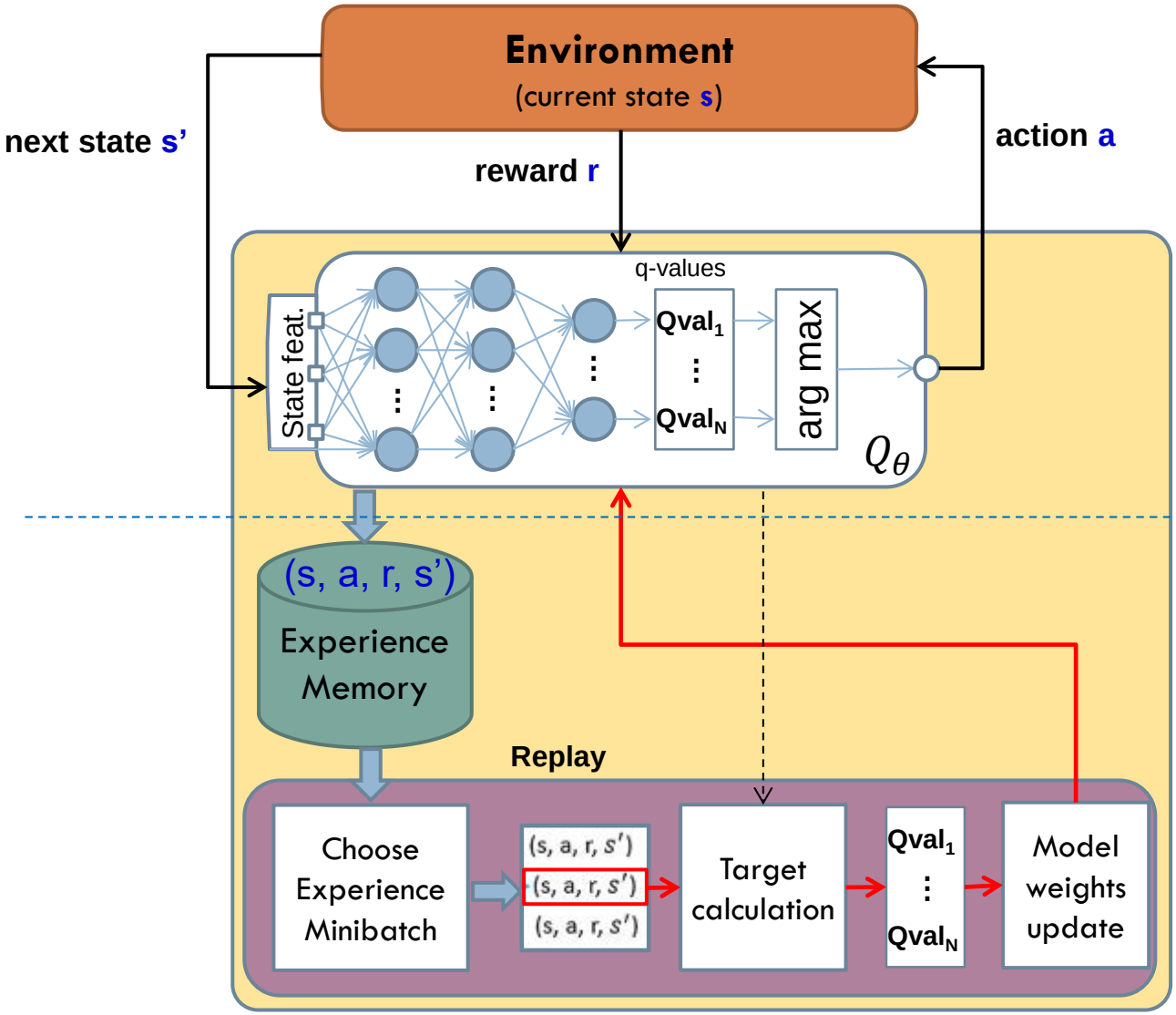
Q Learning



Deep Q Learning

Deep Q-networks

- Estructura de un agente DQN

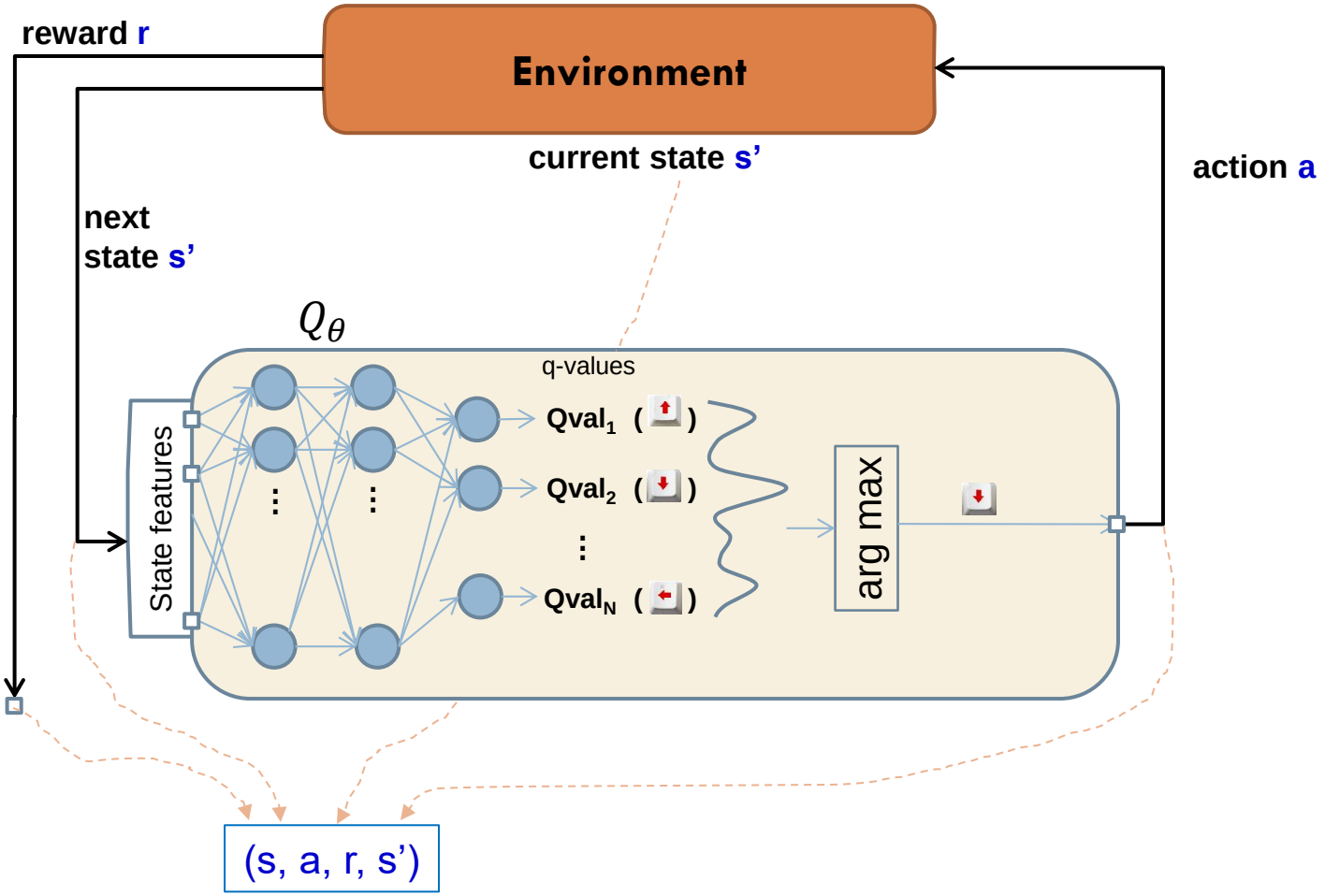


Generación de experiencias

Ajustar modelo (replay)

Deep Q-networks

- Generación de experiencias en DQN



Con cada step del entorno se genera una experiencia que se guarda en la memoria de experiencias

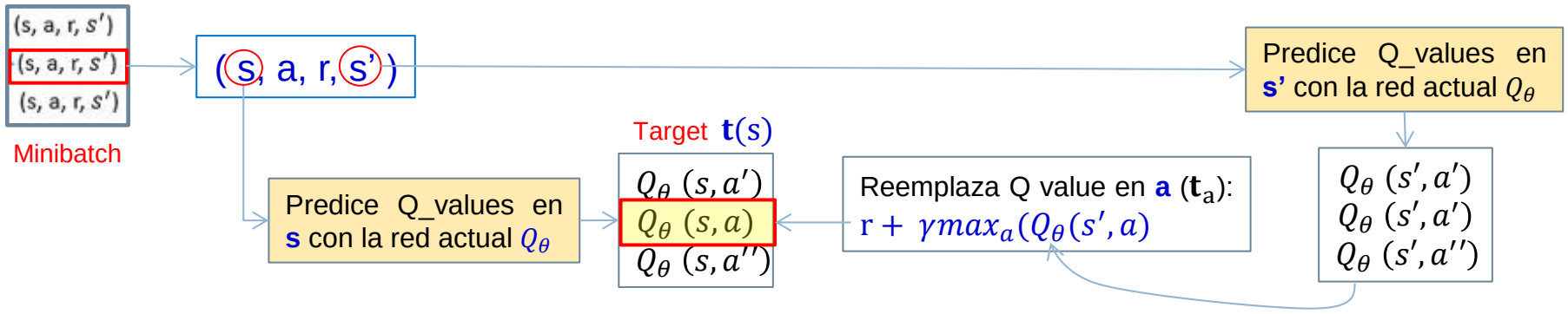
Deep Q-networks

Replay

- Cada step del entorno genera un registro experiencia en la memoria de experiencias:

  (state, action, next_state, reward)

- Cada cierta cantidad de experiencias acumuladas de realiza el replay, donde se reajusta el modelo neuronal (normalmente se realiza al final de cada episodio)
- En un replay se realiza lo siguiente:
  - Minibatch selection: Se selecciona una pequeña muestra de la memoria de experiencias (minibatch)
  - Target calculation: Se calcula los valores Q deseados (**targets**) para cada **state** en minibatch. Estos son los valores predichos por la red actual (Q_θ) en **state**, salvo el valor Q de la acción ejecutada **a**, el cual debe obedecer Bellman (reward + máximo Q predicho en **next_state**):



  - Reajuste del modelo: Se reajusta el modelo actual Q_θ con las tuplas de entrenamiento del minibatch [s, $\mathbf{t}(s)$]

Deep Q-networks

Initialize experience memory D
Initialize model Q_θ with random weights θ
for episode 1:n **do**
    # Make an episode experience with model Q_θ
    s = reset_environment()
    while $s != terminal$
        select an action a
            with probability ε: $a \leftarrow random(Actions(s))$
            otherwise: $a \leftarrow argmax_{a'} Q_\theta(s, a')$
        execute action a in environment and observe reward r and new state s'
        store transition $[s, a, r, s']$ in experience memory D
        $s \leftarrow s'$

    # Update the model Q_θ (replay)
    get a random sample of experiences: $Minibatch \leftarrow Sample(D, batchsize)$
    for each transition $[s, a, r, s'] \in Minibatch$
        $\boldsymbol{t} \leftarrow Q_\theta(s)$ *# vector of q-values predicted by the current model*
        if $s' = terminal$:
            $\boldsymbol{t}_a \leftarrow r$
        else:
            $\boldsymbol{t}_a \leftarrow r + \gamma\, max_{a'} Q_\theta(s', a')$ *# future discounted reward obtained with the model*
        update weights θ of model Q_θ with example $\langle s, \boldsymbol{t} \rangle$
    $\varepsilon \leftarrow \varepsilon * decay$ *# decay the probability of random actions (exploration)*

Double Deep Q-networks

- DQN ajusta el modelo Q_θ con cada experiencia (transición) revisada en el minibatch de replay. El problema es que la construcción del target de entrenamiento se realiza con el mismo modelo que esta siendo ajustado Q_θ : $\boldsymbol{t}_a(s) \leftarrow r + \gamma \, max_{a'} \, Q_\theta(s', a')$

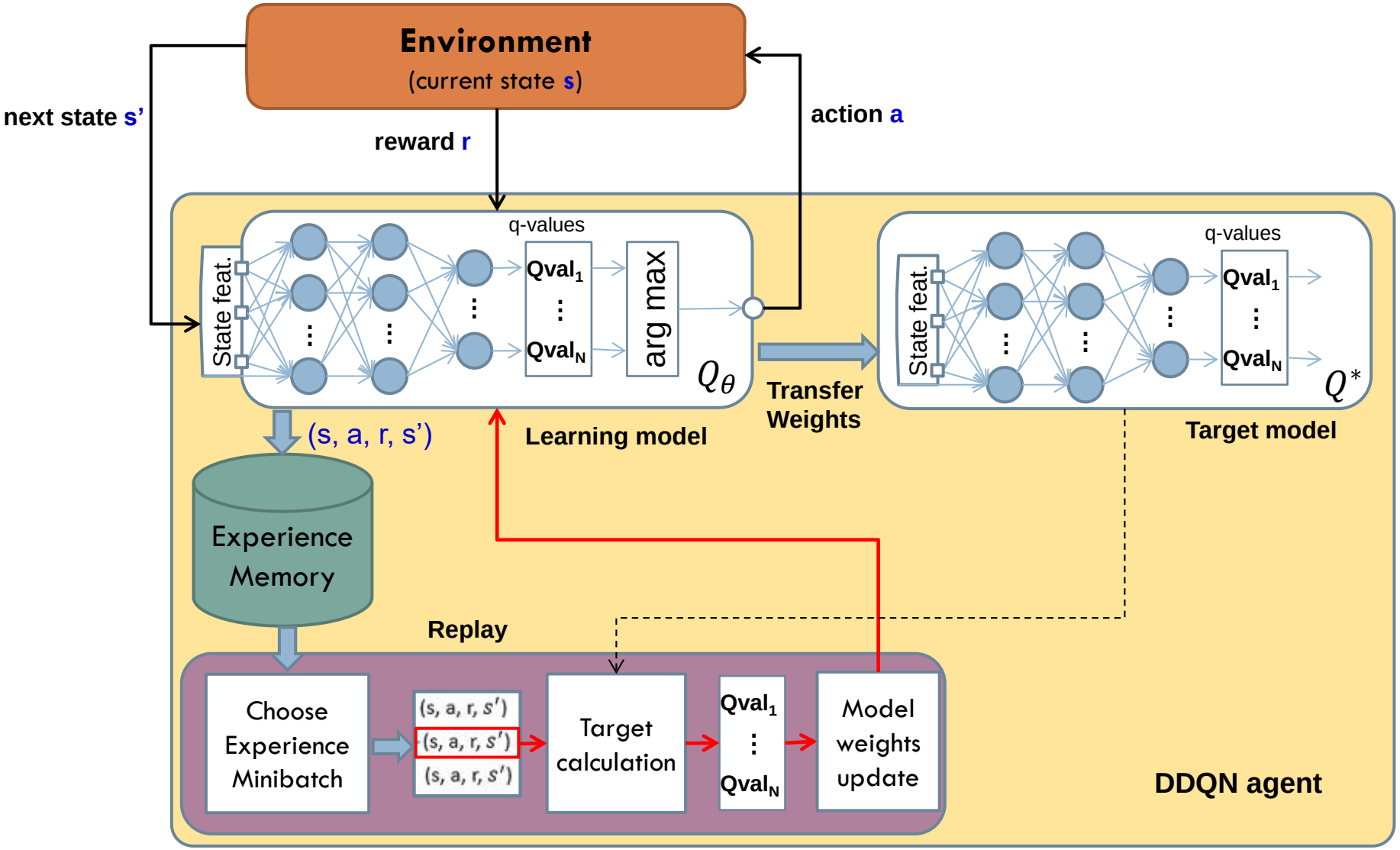
- Ello trae como consecuencia una inestabilidad de aprendizaje en DQN

- Una forma de aliviar ese problema es Double Deep Q-networks, el cual usa un modelo adicional Q^* (target model) para estimar el target durante la etapa de replay: $\boldsymbol{t}_a \leftarrow r + \gamma \, max_{a'} \, Q^*(s', a')$

- Después de cada etapa de replay se transfieren los pesos de Q_θ a Q^* , y este ultimo se usa como estimador de target en el siguiente replay

Double Deep Q-networks

- Flujo de Entrenamiento de un Double DQN (DDQN)



Double Deep Q-networks

Initialize experience memory D
Initialize model Q_θ with random weights θ
Initialize model Q^* with random weights θ^*
for episode 1:n **do**
    # Make a episode experience with model Q_θ
    s = reset_environment()
    while $s != terminal$
        select an action a
            with probability ε: $a \leftarrow random(Actions(s))$
            otherwise: $a \leftarrow argmax_{a'} Q_\theta(s, a')$
        execute action a in environment and observe reward r and new state s'
        store transition $[s, a, r, s']$ in experience memory D
        $s \leftarrow s'$

    # Update the model Q_θ (replay)
    get a random sample of experiences: $Minibatch \leftarrow Sample(D, batchsize)$
    for each transition $[s, a, r, s'] \in Minibatch$
        $\boldsymbol{t} \leftarrow Q_\theta(s)$ *# vector of q-values predicted by the model being learned*
        if $s' = terminal$:
            $\boldsymbol{t}_a \leftarrow r$
        else:
            $\boldsymbol{t}_a \leftarrow r + \gamma\, max_{a'} Q^*(s', a')$ *# future discounted reward obtained with the target model*
            update weights θ of model Q_θ with example $\langle s, \boldsymbol{t} \rangle$
    transfer weights to the target model : $\theta^* \leftarrow \theta$
    $\varepsilon \leftarrow \varepsilon * decay$ *# decay the probability of random actions (exploration)*

Referencias y Material complementar

- DQN: https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf
- DDQN: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12389/11847
- DEMYSTIFYING DEEP REINFORCEMENT LEARNING: https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/
- Deep Reinforcement Learning: Pong from Pixels: https://karpathy.github.io/2016/05/31/rl/
- A Beginner's Guide to Deep Reinforcement Learning: https://skymind.ai/wiki/deep-reinforcement-learning

Preguntas?