El método actor-crítico

PID_00275577

Laura Ruiz Dern

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Laura Ruiz Dern

Doctora en Astronomía y Astrofísica por el Observatoire de Paris. Licenciada en Física y MSc en Astrofísica, Cosmología y Física de Partículas por la Universidad de Barcelona, y doble MSc en Gestión de Proyectos y Cooperación Internacional por la ESNECA Business School. Investigadora y científica de datos con experiencia en *data mining* y *machine learning*, y actualmente especializada en *deep learning* y *deep reinforcement learning*. Ha trabajado en varios proyectos de investigación en astrofísica y ciencia de datos del Centre National d'Études Spatiales (CNES); del Centre National de la Recherche Scientifique (CNRS), en colaboración con la European Space Agency (ESA), y del Centre Tecnològic de Catalunya (Eurecat). Desde 2014 compagina sus investigaciones con la divulgación científica y con la docencia universitaria (Observatoire de Paris, Universitat Oberta de Catalunya).

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Casas Roma

Primera edición: septiembre 2021

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Laura Ruiz Dern
Producción: FUOC

Índice

Introducción

En los anteriores módulos hemos estudiado métodos basados en la función de valor y métodos basados en la política. Como hemos visto, los primeros permiten obtener un valor (recompensa) para cada acción tomada y utilizarla para mejorar las siguientes selecciones; y los segundos permiten definir una política parametrizada que guía el comportamiento del agente sin necesidad de recibir el valor de cada acción.

En este módulo queremos estudiar la combinación de ambos métodos, en búsqueda de un algoritmo híbrido aún más eficiente y estable. Los métodos actor-crítico (AC) son el resultado de esta simbiosis. A nivel conceptual y en la práctica veremos que este algoritmo realiza pequeñas modificaciones respecto de los métodos de gradiente de política, pero es uno de los más potentes en DRL.

Objetivos

Los principales objetivos que estudiaremos en este módulo son:

1. Comprender el origen y la motivación de los métodos AC.

2. Entender la arquitectura y el proceso de aprendizaje de los métodos AC.

3. Estudiar las variaciones A2C y A3C.

4. Descubrir el estado del arte de los métodos AC.

1. Introducción a los métodos actor-crítico

A lo largo de este texto hemos visto, principalmente, dos tipos de métodos para resolver problemas de aprendizaje por refuerzo: los métodos basados en la función de valor (entre los que destacamos el *Q-learning* y las DQN) y los métodos basados en la política (los gradientes de política, con especial mención al método REINFORCE).

Esta distinción viene dada según el tipo de agente, como veíamos al principio. Si el agente está basado en la función de valor, solo tiene en cuenta el valor de las acciones, seleccionando en cada estado aquella de mayor valor; no hay una política de la que el agente sea consciente, aunque esta esté implícita (en realidad, siempre ejecuta la política óptima).

Si el agente está basado en la política, lo que conoce el agente es una representación de la política a seguir, y realiza la selección de las acciones según la política, sin almacenar o usar directamente el valor de esas acciones.

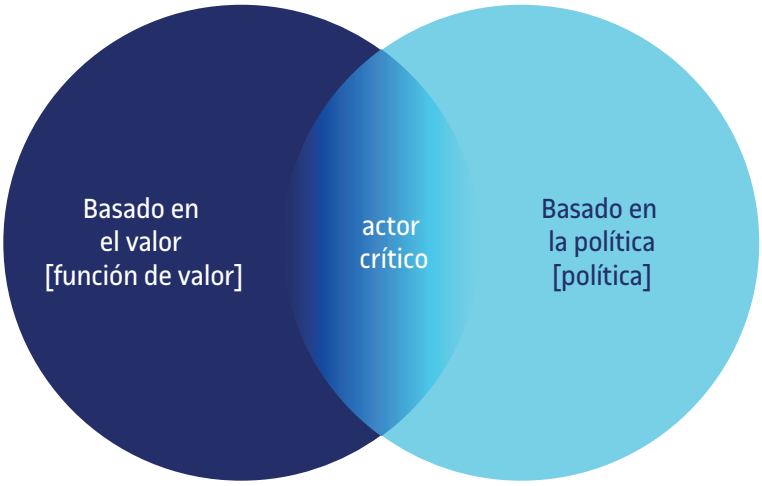
Ambos métodos tienen sus ventajas: los basados en política son mejores en espacios continuos y estocásticos, y los basados en valor suelen ser más eficientes y estables. De aquí que surja la usual y lógica pregunta: ¿no podríamos combinarlos aportando lo mejor de cada uno y salvando así sus propios inconvenientes?

Así es. Si recordamos, en la presentación de la taxonomía de agentes también se introducía un tercer tipo: el **actor crítico** (en inglés *actor-critic*, AC). Este tipo de agente combina, en efecto, los dos anteriores de modo que emplea tanto la función de valor como la política para seleccionar la mejor acción en cada estado (figura 1). La idea es dividir el modelo en dos: uno que proporcione la acción según el estado, y el otro que calcule el valor de Q de esa acción. En la práctica esta combinación se basa en usar dos redes neuronales, una corresponderá al actor y la otra al crítico:

- **Actor**: decide qué acción tomar en cada estado. La función de aproximación será una función basada en la política encargada de guiar el comportamiento del agente en cada uno de los estados, eligiendo la acción que dicte la política.

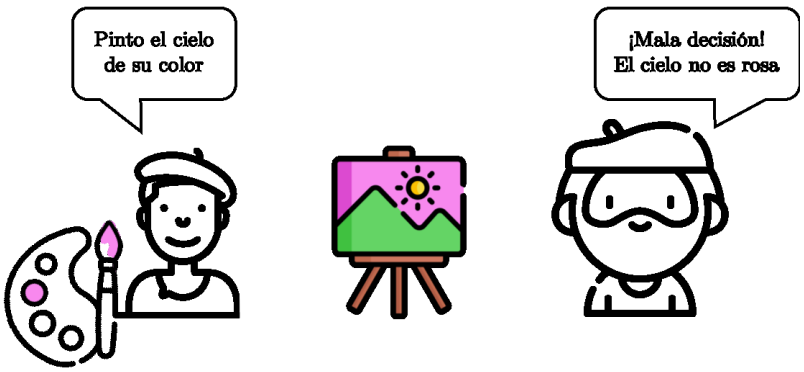
- **Crítico**: evalúa lo buena o mala que ha sido la acción seleccionada para ese estado en concreto. La función de aproximación será una función basada en valor que permitirá calcular el valor de Q para dicha acción.

Figura 1. Esquema visual de la relación entre los métodos basados en la función de valor, los basados en la política, y los híbridos



Para ilustrar este nuevo método, imaginemos un pintor y un joven aprendiz. El joven aprendiz sería el actor y el pintor el crítico. El joven inicialmente sabe poco de pintar, por lo que los primeros días usará los pinceles equivocados, mezclará colores que no encajan, cogerá lienzos que no corresponden, guardará los utensilios en lugares incorrectos, romperá herramientas, manchará todo más de la cuenta, los dibujos serán imprecisos, etc. El pintor por su lado irá observando cómo se desarrolla el joven; algunas veces le corregirá o reñirá, otras lo felicitará (es decir, proporciona el valor de Q, la recompensa, para cada acción). Así, el aprendiz, según lo que vaya diciendo el pintor, irá corrigiendo sus fallos e irá mejorando poco a poco (es decir, irá perfeccionando su política). Gracias a este *feedback* constante, cada vez sabrá mejor qué está bien y qué no (e.g. qué colores se mezclan y cuáles no, cómo se guardan cada uno de los utensilios, cómo hacer un cuadro más realista, qué tipo de pintura usar para cierto tipo de cuadro, etc.). Con el tiempo el joven aprendiz acabará siendo un experto en el arte de pintar.

Figura 2. Ilustración del ejemplo de AC. El actor (aprendiz) toma una acción, y el crítico (pintor) la critica, positiva o negativamente, para que el actor aprenda



Vemos que tanto el pintor como el aprendiz ajustan sus acciones y respuestas con el paso del tiempo. Ambos modelos, el actor y el crítico, participan del proceso de aprendizaje en el cual los dos se adaptan y mejoran en su propio rol. La combinación de ambos, el conjunto, hace que este aprendizaje sea

mucho más eficiente que los dos métodos por separado (el método basado en la función de valor y el método basado en la política), proporcionando una mayor estabilidad y velocidad de convergencia.

1.1 Origen de los métodos AC

En realidad, la idea de combinar estos dos métodos surge originalmente de ciertos inconvenientes que encontramos con los PG, como se ha explicado anteriormente. Es por este motivo que muchas veces se incluye los métodos AC dentro del grupo de los PG. En PG nos encontramos en un escenario Montecarlo. Si recordamos, con el método REINFORCE, hasta que no se terminaba el episodio no obteníamos ninguna recompensa. Esa recompensa se basaba en la recompensa acumulada obtenida de todas las acciones realizadas a lo largo del episodio, a la que llamábamos retorno, G_t. Independientemente de si alguna acción tomada fue mala, si la mayoría fueron buenas, la recompensa acumulada recibida por el conjunto será alta. Pero para ello necesitamos muchas experiencias con tal de conseguir una política óptima. Esto puede conducir a una alta varianza e inestabilidad, que se traduce en un aprendizaje mucho más lento, y más tiempo en converger.

Para reducir esta varianza, añadíamos una línea de base o *baseline*, $b(s)$, al algoritmo de REINFORCE, sustrayendo así la recompensa acumulada. Esta línea de base (cualquier función o valor independiente de la acción) permitía generalizar el algoritmo (el *baseline* puede ser cero), minimizando sustancialmente la varianza (y con ello la velocidad de aprendizaje) sin modificar el valor esperado de la recompensa. Un posible *baseline* era la estimación del valor del estado $\hat{v}_\pi(S_t,w)$.

$$\Delta J(\theta) \approx \mathbb{E}_T \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t,\theta) \left(G_t - b(s_t)\right) \right] \tag{1}$$

Fijémonos que usábamos una función de valor de estado, por lo que (ya entonces) combinábamos una política con una función de valor. Pero es importante puntualizar que en ese caso no estábamos aún en un contexto actor-crítico, porque la función $\hat{v}_\pi(S_t,w)$ actuaba únicamente de línea de base, pero en ningún momento hacía la función de crítico. El crítico actualiza constantemente su respuesta a la acción tomada en cada estado, en cada paso (actualización *online*), mientras que con el *baseline* seguimos en un entorno de Montecarlo en el que no se actualiza nada hasta que el episodio no ha terminado.

En consecuencia, la principal razón de ser de los métodos AC es justamente modificar el proceso de aprendizaje del REINFORCE con *baseline* utilizando métodos de aprendizaje por diferencia temporal (TD), que nos permiten actualizar en cada paso. En pocas palabras podemos decir que los métodos AC son métodos REINFORCE pero usando aprendizaje TD en lugar de Montecarlo.

1.2 Proceso de los métodos AC

Dado que ahora queremos que la actualización se realice de forma continua tras cada paso, necesitamos algo que reemplace la recompensa total que obteníamos al final de cada episodio:

$$\Delta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{N}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})\right]$$

$$= \mathbb{E}_T\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)G(t)\right] \quad (2)$$

porque esta ya no nos interesa como tal. Y aquí entra el rol del crítico. Es decir, en lugar de esta recompensa total lo que queremos es utilizar otra red neuronal que nos aproxime una función de valor de acción como las del *Q-learning*, $Q(s_t,a_t)$, en cada tupla estado-acción, de modo que nos diga si actuamos bien en dicho estado o no. Veamos qué ocurre si descomponemos la esperanza de la anterior ecuación:

$$\Delta J(\theta) \approx \mathbb{E}_{s_0,a_0,\ldots,s_t,a_t}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right]\mathbb{E}_{r_{t+1},s_{t+1},\ldots,r_T,s_T}[G(t)] \quad (3)$$

Podemos ver que el último término es justamente la función de valor $Q(s_t,a_t)$:

$$\mathbb{E}_{s_0,a_0,\ldots,s_t,a_t}[G(t)] = Q(s_t,a_t) \quad (4)$$

Luego, podemos reescribir la función de actualización como:

$$\Delta J(\theta) \approx \mathbb{E}_{s_0,a_0,\ldots,s_t,a_t}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right]Q(s_t,a_t)$$

$$= \mathbb{E}_T\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)Q(s_t,a_t)\right] \quad (5)$$

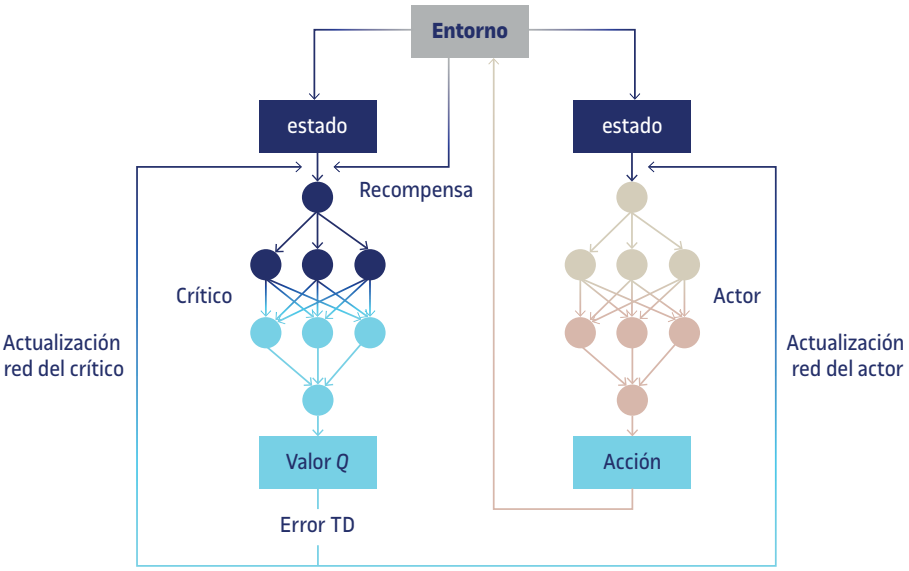
donde el término $\pi_\theta(a_t|s_t)$ (la política) es la red neuronal del actor, es decir, el que selecciona las acciones; y el término $Q(s_t,a_t)$ es la red neuronal del crítico, es decir, el que critica las acciones tomadas por el actor. Calculando esta función de valor en cada paso (TD) podremos mejorar el aprendizaje.

Este proceso es *on-policy*, ya que el crítico debe aprender y criticar la política que emplee el actor en cada uno de los estados, por lo que se trata de un aprendizaje continuo.

La figura 3 ilustra gráficamente el algoritmo AC, en el que se muestra cómo tanto el actor como el crítico adquieren la información del entorno y compu-

tan los parámetros con sendas redes neuronales. La acción elegida por el actor interacciona con el entorno y se obtiene una recompensa. La salida proporcionada por el crítico servirá finalmente para actualizar ambas redes guiando así el aprendizaje del agente.

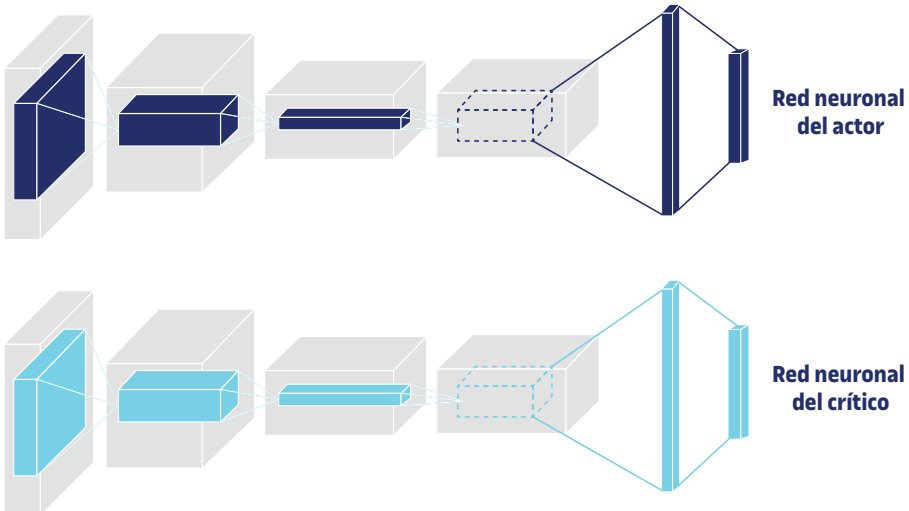
Figura 3. Esquema del algoritmo AC



Para entrenar un método AC podemos optar por dos tipos de arquitectura distintos:

1) Diseñar y entrenar dos redes totalmente por separado y en paralelo, una para el actor y la otra para el crítico (figura 4). Las dos redes usarán ascenso de gradiente (nuestro objetivo es encontrar el máximo global) para actualizar sus pesos en cada paso. El método implica más parámetros pero también más estabilidad en el entrenamiento.

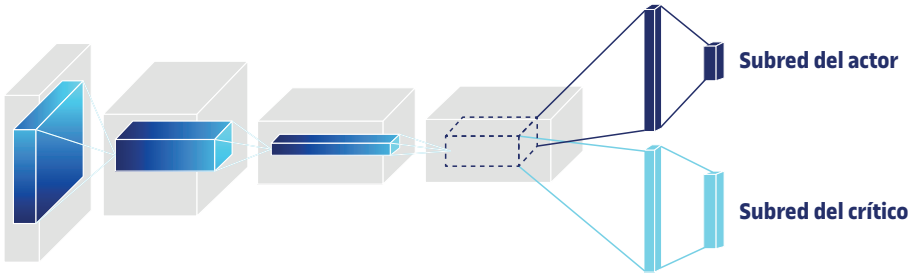
Figura 4. Arquitectura del método AC con dos redes separadas, para el actor y el crítico



2) Entrenar una única red que luego se divide en dos subredes (figura 5), pa-

recido a la arquitectura de las *dueling* DQN. Dado que comparten toda la primera parte de la arquitectura, eso implicará muchos menos parámetros y una simplificación del proceso (por ejemplo, filtros convolucionales, etc.) pero, a la vez, no siempre será fácil encontrar un coeficiente que equilibre de forma balanceada la pérdida del actor y del crítico conjuntamente.

Figura 5. Arquitectura del método AC con un primer bloque compartido que se divide en dos subredes para el actor y el crítico



La entrada de las dos redes (primer tipo) o de la red común (segundo tipo), siempre es el estado actual S_t. Durante el proceso, en cada paso t, la política (actor) proporciona una acción A_t a la función de valor (crítico), y esta le devuelve una recompensa (valor Q). Luego, junto con el nuevo estado S_{t+1}, el actor actualiza sus pesos para proporcionar la siguiente acción A_{t+1} al crítico, para que este actualice también sus pesos. Y así sucesivamente, como se mostraba en el esquema de la figura 3.

El Algoritmo 1 muestra el método AC para la función de valor de acción Q.

Algorithm 1 Actor-crítico para la función Q

Inicializar parámetros s, θ, w, α_θ y α_w (*learning rates*)
Inicializar distribución $a \sim \pi_\theta(a|s)$
for $t = 0,1,...T$ **do**
  Muestrear la recompensa $r_t \sim R(s|a)$ y el siguiente estado $s' \sim P(s'|s,a)$
  Muestrear la siguiente acción $a' \sim \pi_\theta(a'|s')$
  Actualizar los parámetros de la política:
    $\theta \leftarrow \theta + \alpha_\theta Q_w(s,a)\nabla_\theta \log \pi_\theta(a|s)$
  Calcular la corrección (error TD) para el valor de la acción en el instante t:
    $\delta_t = r_t + \gamma Q_w(s',a') - Q_w(s,a)$
  Actualizar los parámetros de la función Q:
    $w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s,a)$
  Actualizar $a \leftarrow a'$ y $s \leftarrow s'$
end for

2. Actor-crítico con ventaja (A2C)

En el algoritmo anterior estamos calculando el valor de Q para cada una de las acciones que el agente toma, es decir, el crítico da una opinión sobre cada acción. Pero si recordamos la motivación de las *dueling* DQN en el apartado anterior, ¿realmente necesitamos calcular cada una de las Q? ¿No podría haber estados en los que ya sepamos por adelantado que ninguna acción aportará cambios significativos? En tal caso no sería necesario hacer ese cálculo, reduciríamos la variabilidad y mejoraríamos el rendimiento. ¿Podemos aplicar esta idea en los métodos AC?

La solución a este problema en las *dueling* DQN pasaba por desacoplar la función de Q en dos términos: el valor de estar en un estado s, $V(s)$, más la ventaja de tomar la acción a en ese estado s, $A(s,a)$:

$$Q(s,a) = A(s,a) + V(s) \tag{6}$$

En los métodos AC tiene sentido también hacer esta sustitución, puesto que estados diferentes podrían tener diferentes líneas de base. Así podríamos hacer que la línea de base fuera directamente dependiente del estado, reduciendo aún más la varianza. De modo que:

$$A(s,a) = Q(s,a) - V(s) \tag{7}$$

Es decir, si $Q(s,a)$ es la recompensa esperada por tomar la acción a, $A(s,a)$ sería como la recompensa extra que obtenemos al tomar esa acción. Si $A(s,a) < 0$ significará que la acción elegida no es buena y nos da un valor inferior a lo esperado en ese estado. El gradiente se inclinará hacia una dirección u otra según sea el valor de la ventaja (es decir, en dirección contraria cuando la ventaja sea negativa).

Fijémonos que con este desacoplamiento de Q, la ecuación 5 la podríamos escribir de la siguiente manera:

$$\Delta J(\theta) \approx \mathbb{E}_T \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(Q(s_t,a_t) - V(s_t)\right) \right] \tag{8}$$

Es decir, la función $V(s)$ estaría actuando de *baseline*. Una línea de base que solo depende del estado s; justo lo que buscábamos para poder reducir la varianza.

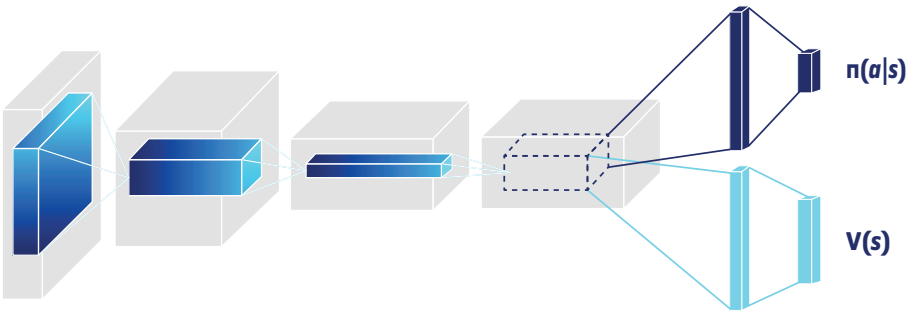
El problema es que ahora no solo tenemos la red neuronal de la política y la de la función Q (de la cual queríamos librarnos de tener que calcular), sino que acabamos de añadir una tercera, la de V. Tres redes es, ciertamente, muy ineficiente. Para evitarlo podemos servirnos de la relación que tienen Q y V mediante la ecuación de Bellman:

$$Q(s_t,a_t) = \mathbb{E}\left[r_{t+1} + \gamma V(s_{t+1})\right] \tag{9}$$

donde, recordemos, r es la recompensa actual y γ el factor de descuento. De modo que, sustituyendo en la ecuación 8:

$$\Delta J(\theta) \approx \mathbb{E}_T\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\,(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))\right] \tag{10}$$

Figura 6. Esquema de la red neuronal A2C



donde el término $r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ no es más que el error en las diferencias temporales (TD) al que nos referíamos en la figura 3 del apartado anterior. Este término sigue siendo la ventaja $A(s_t,a_t)$, pero parametrizada únicamente con la función V, por lo que volvemos a tener solo dos redes neuronales: la de la política que ya teníamos y la de V (figura 6), simplificando sustancialmente el cálculo. Este algoritmo es el llamado actor-crítico con ventaja, en inglés ***advantage actor-critic***, más conocido por su acrónimo **A2C**.

Nota

Dado que A2C es el algoritmo estándar que se suele utilizar, en muchos casos referirse al método AC significa el A2C por defecto.

Ahora el crítico, en lugar de aprender los valores Q, aprenderá los de la ventaja. De esta manera, su evaluación sobre una acción dada se basará no solo en lo buena o mala que ha sido su elección, sino también en cuánto mejor puede ser. Este cambio permite reducir la alta variabilidad que suele tener la red neuronal de la política y mejorar la estabilidad de todo el modelo.

3. Actor-crítico asíncrono con ventaja (A3C)

En el apartado de las DQN vimos que uno de los principales problemas en DRL es la fuerte correlación entre los estados. Entonces lo resolvíamos introduciendo un *buffer* de repetición (*replay buffer*). Este nos permitía almacenar las experiencias pasadas pasando a la red neuronal solo un subconjunto aleatorio de estas, rompiendo así un poco la correlación. Más adelante veíamos también una mejora con el método de priorización de experiencias (PER), con el que dábamos más importancia a aquellas experiencias más significativas, ajustando unos pesos a todas las experiencias para evitar un sesgo hacia las más prioritarias.

En los métodos basados en la política, de los que hemos visto que derivan los métodos AC, existe el mismo problema de correlación entre estados. Utilizar un *buffer* de repetición de experiencias consume mucha memoria, así que una alternativa para los métodos AC es entrenar paralelamente muchos agentes en múltiples entornos, donde cada uno de estos agentes no es más que una copia de la misma red neuronal (con la misma configuración de la figura 5). Estos agentes, periódicamente y de forma asíncrona e independiente, van actualizando la función de valor global. Después de cada actualización, los agentes actualizan sus parámetros con los de la red global y continúan su proceso de aprendizaje por n pasos hasta la siguiente actualización. Así, dado que cada vez que un agente se actualiza está pasando su información a la red neuronal global, en realidad la actualización de sus parámetros está teniendo en cuenta ya la información aportada por los demás agentes hasta ese momento. Todos se alimentan de la aportación de todos, y todos contribuyen a mejorar la red global. La figura 7 ilustra este procedimiento de manera más gráfica. Este método, introducido por Mnih, Puigdomènech Badia, Mirza, *et al*. (2016), se llama actor-crítico asíncrono con ventaja, en inglés ***asynchronous advantage actor-critic***, más conocido por su acrónimo **A3C**.

En A2C también podemos tener muchos agentes entrenando a la vez, pero no es hasta que todos terminan su proceso de aprendizaje (tras n pasos) que se sincronizan todos sus parámetros y la red completa se actualiza y con ella los parámetros de todos los agentes a la vez. En la figura 8 podemos observar la sutil diferencia respecto de la figura 7. Es por eso que al método A2C también se le llama el método síncrono, en contraste con el asíncrono A3C. Como se puede intuir, trabajar con actores asíncronos permite, en general, explorar el espacio de estados de forma más efectiva y eficiente, y en mucho menos tiempo porque pueden cubrir mucho más. Pero, por contra, dado que no se sincronizan a la vez, algunos de los agentes pueden estar aprendiendo aún

Lectura complementaria

V. Mnih; A. Puigdomènech Badia; M. Mirza, *et al.* (2016). *Asynchronous Methods for Deep Reinforcement Learning*. Google DeepMind.

con parámetros desactualizados mientras que otros ya trabajan con nuevos, lo que en el momento de agregar sus resultados a la actualización global, el resultado puede terminar siendo subóptimo. Es por eso que, en muchos casos, A2C acaba proporcionando un rendimiento muy parecido a A3C y puede ser incluso más efectivo.

Figura 7. Esquema del proceso de aprendizaje del método A3C

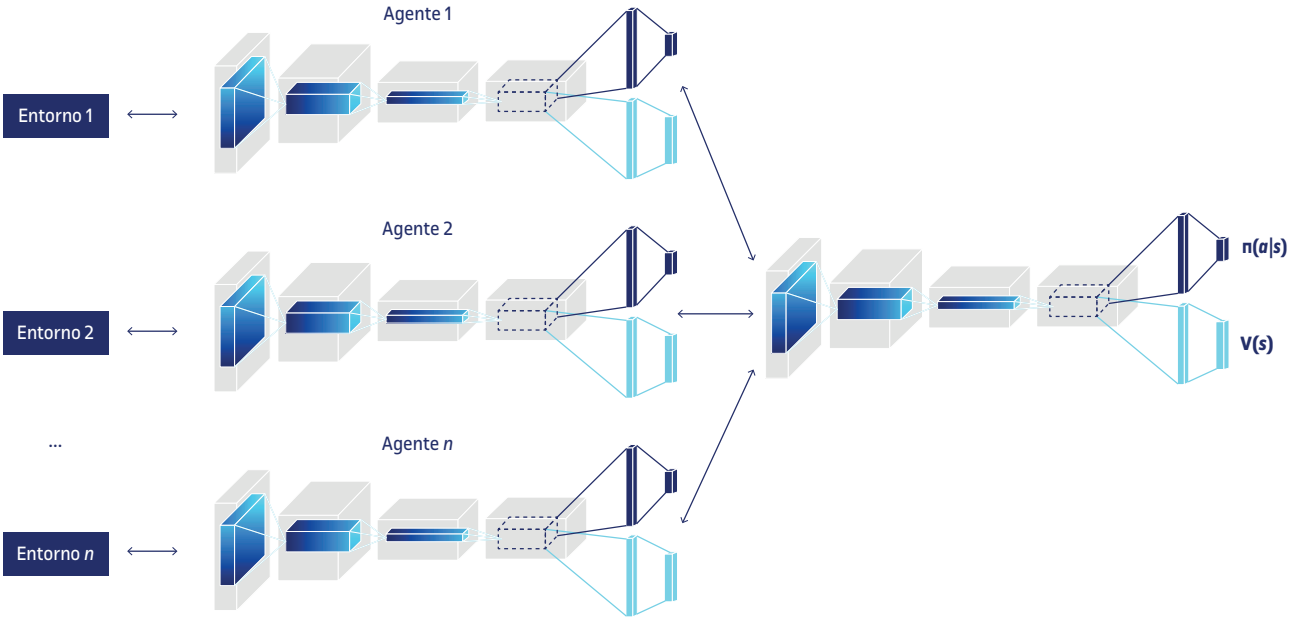
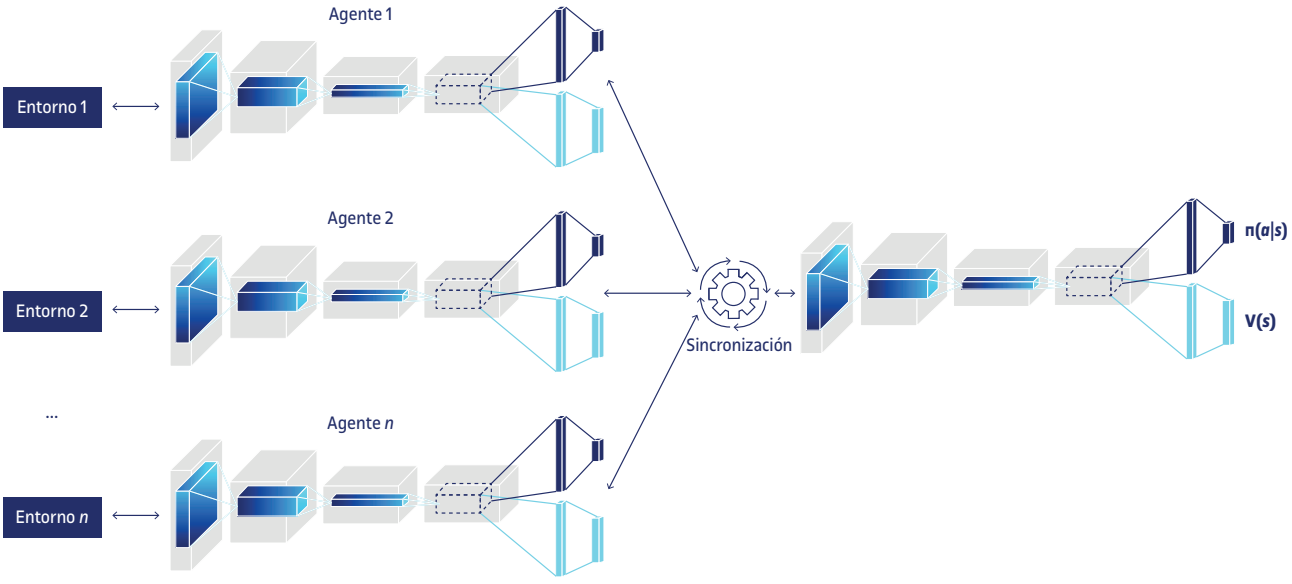


Figura 8. Esquema del proceso de aprendizaje del método A2C con múltiples agentes



El Algoritmo 2 muestra el proceso del método A3C paso a paso para el caso de la función de valor de acción Q.

Algorithm 2 A3C para la función Q

Conocidos los parámetros globales θ, los de la red objetivo θ^- y los específicos θ' de cada actor:
Inicializar tiempo $t = 1$
Inicializar parámetros red objetivo $\theta^- \leftarrow \theta$
Inicializar parámetros específicos de los actores $\theta' = \theta$
Inicializar gradientes de la red $d\theta \leftarrow 0$
while episodio $T \geq T_{\text{MAX}}$ **do**
  Reinicializar gradientes $d\theta = 0$
  Sincronizar los parámetros específicos con los globales $\theta' = \theta$
  Establecer $t_{\text{inicio}} = t$
  Obtener un s_t inicial
  while s_t! = terminal and $t - t_{inicio} \leq t_{\max}$ **do**
    Seleccionar una acción según la política ϵ-*greedy* basada en $Q_w(s_t,a)$
    Obtener recompensa r_t y nuevo estado s_{t+1}
    Actualizar $t \leftarrow t + 1$ y $T \leftarrow T + 1$
  end while
  Inicializar el retorno R:
  if s_t terminal **then**
    $R = 0$
  else
    $R = \max_a Q(s_t,a;\theta^-)$
  end if
  for $i = t-1,\ldots,t_{\text{inicio}}$ **do**
    $R \leftarrow \gamma R + r_i$
    Acumular los gradientes:
      $\theta' : d\theta \leftarrow d\theta + \frac{\partial (R - Q(s_i,a_i;\theta'))^2}{\partial \theta'}$
  end for
  Actualizar asíncronamente θ usando $d\theta$
  if $T \bmod I_{\text{objetivo}} == 0$ **then**
    $\theta^- \leftarrow \theta$
  end if
end while

4. Taxonomía de algoritmos *model-free* en DRL

En este módulo hemos visto en qué consisten los métodos AC y dos de sus variantes más conocidas, el A2C y el A3C. Aun así, existen multitud de variantes y algoritmos que buscan combinar los algoritmos *model-free* estudiados hasta ahora, variar enfoques, o cambiar parámetros, siempre con el fin de mejorar rendimiento y eficiencia. No forma parte del objetivo de este texto entrar en el detalle de todos estos algoritmos porque son muchos y no sería posible abordarlos todos de la forma que merecen, pero sí resulta interesante mencionarlos para darse cuenta del vasto y extenso mundo del aprendizaje por refuerzo profundo, y todas las posibilidades que ofrece.

Entre estos algoritmos *model-free* podemos encontrar:

- **Gradiente de política determinista** (*deterministic policy gradient*, **DPG**, Silver, Lever, Heess, *et al.*, 2014).

  Modela la política como una decisión determinista, no como una distribución de probabilidad.

- **Gradiente de política determinista profundo** (*deep deterministic policy gradient*, **DDPG**, Lillicrap, Hunt, Pritzel, *et al.*, 2015).

  Combina el método DPG con la DQN clásica en un espacio continuo y una política determinista.

- **DDPG distribucional distribuido** (*Distributed Distributional* DDPG, **D4PG**, Barth-Maron, Hoffman, Budden, *et al.*, 2018).

  Modifica el algoritmo DDPG de modo que el crítico estima el valor de la función de valor a partir de una función de distribución parametrizada por los pesos.

- **Multiagente DDPG** (*multi-agent* DDPG, **MADDPG**, Lowe, Wu, Tamar, *et al.*, 2017).

  Extensión del algoritmo DDPG a múltiples agentes coordinados para realizar tareas de forma independiente (sin saber las políticas de los demás agentes) únicamente con información local.

- **Retraso parejo del modelo DDPG** (*twin delayed deep deterministic*, **TD3**, Fujimoto, Van Hoof, Meger, *et al.*, 2018).

  Partiendo del modelo DDPG, reduce la frecuencia de actualización de la política respecto de la actualización de la función Q, parecido al funcionamiento de la red objetivo en las DQN, con el objetivo de reducir la varianza.

- **Gradiente de política con variación de Stein** (*Stein variational policy gradient*, **SVPG**, Liu, Ramachandran, Liu, *et al.*, 2017).

  Implementa la variación del gradiente de Stein para actualizar un parámetro aleatorio de la política $\theta \sim q(\theta)$, con el objetivo que sea $q(\theta)$ la distribución a aprender.

- **Optimización de la política en región de confianza** (*trust region policy optimization*, **TRPO**, Schulman, Levine, Moritz, *et al.*, 2015).

  Tiene como objetivo maximizar la función objetivo tal que la distancia (divergencia KL) entre políticas anteriores y nuevas sea suficientemente pequeña para ayudar a mantener la estabilidad del modelo.

- **Optimización de la política de proximidad** (*proximal policy optimization*, **PPO**, Schulman, Wolski, Dhariwal, *et al.*, 2017).

  Simplificación del método TRPO restringiendo dentro de un intervalo la distancia posible entre la política anterior y la nueva, con tal de evitar la actualización de la política tras largos episodios.

- **Actor-crítico con experiencia de repetición** (*actor-critic with experience replay*, **ACER**, Wang, Bapst, Heess, *et al.*, 2016).

  Versión *off-policy* del método A3C usando experiencia de repetición para reducir más la correlación entre estados.

- **Actor-crítico usando una región de confianza de factorización Kronecker** (*actor-critic using Kronecker-factored trust region*, **ACKTR**, Wu, Mansimov, Liao, *et al.*, 2017).

  Cambia la manera de hacer la actualización del gradiente para el actor y el crítico usando una aproximación factorizada de Kronecker.

- **Actor-crítico suave** (*soft actor-critic*, **SAC**, Haarnoja, Zhou, Abbeel, *et al.*, 2018).

  Método *off-policy* que incorpora la medida de entropía de la política en la recompensa con el objetivo de favorecer la estabilidad y la exploración.

- **Repetición de experiencia retrospectiva** (*hindsight experience replay*, **HER**, Andrychowicz, Wolski, Ray, *et al.*, 2017).

  Es una técnica de repetición de experiencias (como el PER que hemos visto en este texto) que permite un aprendizaje más eficiente de los algoritmos de RL en casos o comportamientos complejos en los que las recompensas son binarias o no usuales. La idea tras esta técnica es la de imaginar qué pasaría si las circunstancias fueran diferentes (inspirado en la idea humana que de los errores se aprende): se fija inicialmente un objetivo pero se considera que el agente no lo consigue y termina en otro estado distinto al esperado; en lugar de recompensar negativamente por ello, se asume que este nuevo estado era en realidad el objetivo esperado (de una trayectoria imaginaria), devolviendo siempre recompensas positivas.

- **Regresión cuantil DQN** (*quantile regression DQN*, **QR-DQN**, Dabney, Rowland, Bellemare, *et al.*, 2017).

  En lugar de promediar la distribución probabilística de estados, acciones y recompensas, como vimos en el algoritmo DQN categórica (C51), este método modela directamente la distribución de las recompensas, disminuyendo así la aleatoriedad de la recompensa observada a largo plazo.

- **Redes cuantiles implícitas** (*implicit quantile networks*, **IQN**, Dabney, Hoffman, Budden, *et al.*, 2018).

  Implementa una mejora más flexible del algoritmo QR-DQN usando una regresión cuantil para aproximar la función cuantil global para la distribución del retorno, con el objetivo de romper la restricción de las QR-DQN de que las distribuciones de probabilidad tengan que ser uniformes para todos los cuantiles.

- **Función cuantil completamente parametrizada** (*full parametrized quantile function*, **FQF**, Yang, Zhao, Lin, *et al.*, 2019).

  Los algoritmos C51, QR-DQN e IQN parametrizan o bien la probabilidad o bien el valor del retorno de la función de distribución (ya sea de forma uniforme o aleatoria). Este nuevo método permite parametrizar ambos lados, tanto el de los cuantiles como el de los valores, usando dos redes neuronales entrenadas a la vez: una que genera el conjunto de fracciones de cuantiles para cada tupla estado-acción, y otra que mapea probabilidades a valores de cuantiles.

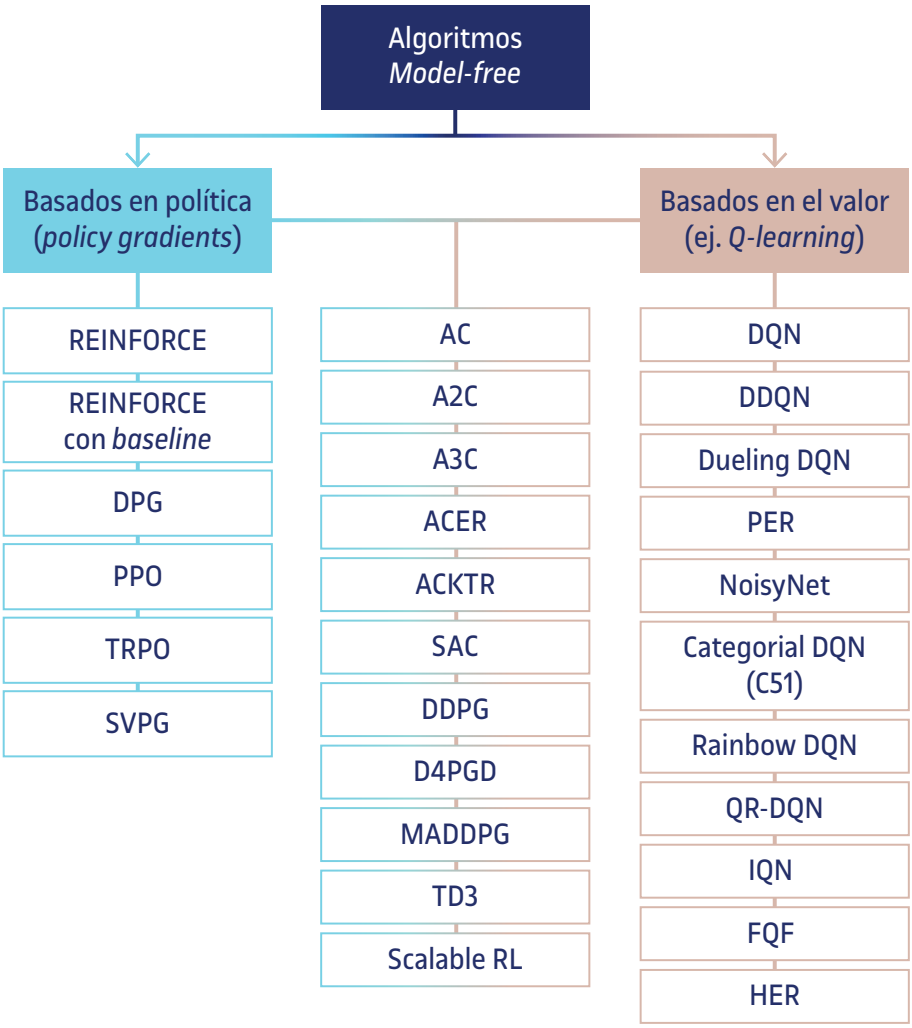
- **Agentes de RL escalables** (*scalable RL agents*).

  Tienen como objetivo conseguir entrenar a un único agente en múltiples tareas (siendo el número de tareas muy elevado). Para ello separan las acciones y la adquisición de experiencias del aprendizaje en sí, de modo que

muchos actores generan experiencias mientras que el agente aprende y optimiza la política y la función de valor a partir de todas estas experiencias a la vez. Algunos algoritmos *model-free* de este tipo son IMPALA (*importance weighted actor-learner architecture*, Espeholt *et al.*, 2018), R2D2 (*recurrent replay distributed* DQN, Kapturowski, Ostrovski, Quan, *et al.*, 2019) o SEED (*scalable efficient deep-RL*, Espeholt, Marinier, Stanczyk, *et al.*, 2019).

La figura 9 muestra esquemáticamente la clasificación de todos los algoritmos estudiados en este texto (incluyendo los citados en este apartado) diferenciándolos en los basados en la política, los basados en la función de valor, y los que combinan ambas aproximaciones. Es importante tener presente que se trata de una clasificación propia de este texto no estrictamente bien representada, en la que solo se han tenido en cuenta las características básicas de los algoritmos omitiendo detalles más complejos, dado que es muy difícil realizar un esquema preciso de esta extensa (e incompleta) taxonomía. La intención es proporcionar al lector una visión suficientemente clara, sencilla y global del estado del arte de los principales algoritmos *model-free* del DRL.

Figura 9. Taxonomía (incompleta) de algoritmos libres de modelo en DRL



Resumen

En este módulo hemos visto que el uso simultáneo de una solución aproximada para la política y una para la función de valor nos permite obtener un algoritmo capaz de seleccionar la mejor acción en cada estado gracias al diálogo entre un actor y un crítico, en el que el primero proporciona la acción a tomar y el segundo evalúa la bondad de esta acción elegida (su valor Q).

Dos de las principales mejoras de este algoritmo son el A2C y el A3C. A2C introduce una idea similar a la de las *dueling* DQN desacoplando la función de Q en una función de valor del estado y la ventaja de tomar cierta acción en ese estado. Así, la línea de base depende directamente de cada estado en cuestión y se consigue reducir la varianza. Tanto A2C como A3C permiten que múltiples agentes realicen su proceso de aprendizaje en copias distintas del mismo entorno, paralela e independientemente, pero, a diferencia de A2C, en A3C los agentes sincronizan sus resultados con la red principal de forma asíncrona. De este modo el espacio de estados a explorar puede ser mucho mayor y se puede realizar en menor tiempo.

Este tipo de redes han demostrado mejorar la estabilidad de las soluciones en muchos problemas y conforman la base del estado del arte del DRL. Como hemos visto, existen multitud de otros algoritmos *model-free* en la literatura que buscan –en la combinación de parámetros, cambios de enfoque, y aproximaciones varias– perfeccionar la eficiencia, aplicabilidad y rendimiento en la solución de problemas en aprendizaje profundo. Los algoritmos libres de modelo son los más desarrollados hasta el momento y son los que hemos abordado en este texto. Sin embargo, recordemos que también podemos encontrar algoritmos de DRL *model-based*, que buscan acelerar aún más el proceso de entrenamiento. Brevemente, la idea es introducir un modelo del entorno que permita reutilizar varias veces la experiencia adquirida durante la interacción agente-entorno mejorando así las acciones del agente con respecto al objetivo establecido. Es decir, una función capaz de predecir las transiciones de estado y las recompensas, acotando el rango de mejores opciones para el agente. Los algoritmos I2A o AlphaZero son ejemplos de estos métodos. Aunque estos algoritmos escapan de los objetivos de este texto, mencionarlos nos permite comprender aún mejor el vasto campo del estado del arte del aprendizaje por refuerzo profundo.

Glosario

aprendizaje profundo *m* Conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel utilizando redes neuronales de múltiples capas o niveles.
sigla **DL**
en deep learning

deep learning *m* Véase **aprendizaje profundo**.

DL *m* Véase **aprendizaje profundo**.

i.i.d. *m* Véase **independientes e idénticamente distribuidos**.

independent and identically distributed *m* Véase **independientes e idénticamente distribuidos**.

independientes e idénticamente distribuidos *m* Conjunto de datos o variables aleatorias que son mutuamente independientes entre sí y además cada variable tiene la misma distribución de probabilidad.
sigla **i.i.d.**
en independent and identically distributed

Bibliografía

Andrychowicz, M.; Wolski, F.; Ray, A., *et al.* (2017). *Hindsight Experience Replay*. OpenAI, arXiv:1707.01495v3.

Barth-Maron, G.; Hoffman, M. W.; Budden, D., *et al.* (2018). *Distributed Distributional Deterministic Policy Gradients*. International Conference on Learning Representations.

Dabney, W.; Ostrovski, G.; Silver, D., *et al.* (2018). «Implicit Quantile Networks for Distributional Reinforcement Learning». *PMLR*, 80, arXiv:1806.06923v1.

Dabney, W.; Rowland, M.; Bellemare, M. G., *et al.* (2017). *Distributional Reinforcement Learning with Quantile Regression*. arXiv:1710.10044v1.

Espeholt, L.; Soyer, H.; Munos, R., *et al.* (2018). *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*.

Espeholt, L.; Marinier, R.; Stanczyk, P., *et al.* (2019). *SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference*. Brain Team, Google Research, ICLR, arXiv:1910.06591v2.

Fujimoto, S.; van Hoof, H.; Meger, D. (2018). «Addressing Function Approximation Error in Actor-Critic Methods». *PMLR*, 80, arXiv:1802.09477v3.

Haarnoja, T.; Zhou, A.; Abbeel, P., *et al.* (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. Berkeley: Berkeley Artificial Intelligence Research, University of California, arXiv:1801.01290v2.

Kapturowski, S.; Ostrovski, G.; Quan, J., *et al.* (2019). *Recurrent experience replay in distributed in reinforcement learning*. Google DeepMind, arXiv:1802.01561v3.

Lapan, M. (2018). «Deep Reinforcement Learning Hands-On». *Expert Insight*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A., *et al.* (2015). *Continuous control with deep reinforcement learning*. Google DeepMind.

Liu, Y.; Ramachandran, P.; Liu, Q., *et al.* (2017). *Stein Variational Policy Gradient*. arXiv:1704.02399v1.

Lowe, R.; Wu, Y.; Tamar, A., *et al.* (2017). *Multi-Agent Actor-Critic for MixedCooperative-Competitive Environments*. arXiv:1706.02275v4.

Mnih, V.; Puigdomènech Badia, A.; Mirza, M., *et al.* (2016). *Asynchronous methods for deep reinforcement learning*. International Conference on Machine Learning.

Schulman, J.; Levine, S.; Moritz, P., *et al.* (2015). *Trust region policy optimization*. Proceedings of the 32nd International Conference on Machine Learning. ICML-15.

Schulman, J.; Wolski, F.; Dhariwal, P., *et al.* (2017). *Proximal policy optimization algorithms*. arXiv preprint arXiv:1707.06347.

Silver, D.; Lever, G.; Heess, N., *et al.* (2014). *Deterministic Policy Gradient Algorithms*. ICML, hal-00938992.

Sutton, R. S.; Barto, A. G. (2019). *Reinforcement Learning. An introduction*. Cambridge, MA: MIT Press.

Thomas, S.. *Deep Reinforcement Learning Course*. [Fecha de consulta: 3 de junio de 2020]. <https://simoninithomas.github.io/Deep_reinforcement_learning_Course/>.

Wang, Z.; Bapst, V.; Heess, N., *et al.* (2016). *Sample Efficient Actor-Critic with Experience Replay*. Google DeepMind, arXiv:1611.01224v2.

Weng, L. (2018). *Policy Gradient Algorithms*. [Fecha de consulta: 1 de junio de 2020]. <https://lilianweng.github.io/lil-log>.

Wu, Y.; Mansimov, E.; Liao, S., *et al.* (2017). *Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation*. arXiv:1708.05144v2.

Yang, D.; Zhao, L.; Lin, Z., *et al.* (2019). *Fully Parameterized Quantile Function for Distributional Reinforcement Learning*. arXiv:1911.02140v3.