Universitat d'Alacant
Universidad de Alicante

Robots móviles

Grado en Ingeniería Robótica

Control de robots móviles mediante técnicas de aprendizaje por refuerzo

Trabajo teórico

Autores:
Àngel Alepuz Jerez
Rafael Antón Cabrera
Andrés Gómez-Caraballo Yélamos
Adrián Sanchis Reig

Enero 2022

Índice

1. Introducción

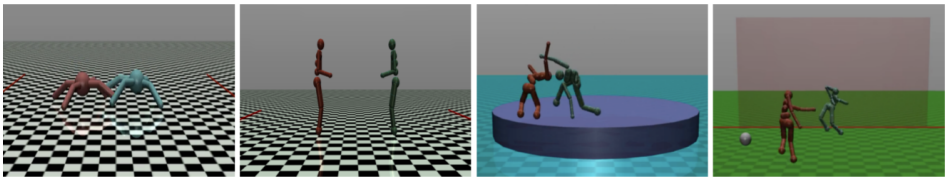
En este trabajo se introduce el uso de técnicas de aprendizaje por refuerzo para el control de robots móviles. El aprendizaje por refuerzo se define como el estudio de agentes inteligentes capaces de aprender a base de prueba y error [1]. Establece las bases de una serie de algoritmos cuya propuesta principal consiste en recompensar positivamente o castigar negativamente el comportamiento de un agente para que este en el futuro sea más propenso a repetir las acciones positivas y rechazar las conductas negativas.

Los métodos de aprendizaje por refuerzo en los últimos años han logrado una gran cantidad de éxitos. Cabe resaltar que estos logros, no se ciñen únicamente al campo de la robótica móvil. El aprendizaje por refuerzo supone un paradigma para resolver multitud de problemas con un enfoque distinto de los métodos tradicionales de programación.

Por ejemplo, cuenta con interesantes aplicaciones en juegos de estrategia y en campos de la robótica como la manipulación de objetos. En estos campos, el uso de aprendizaje por refuerzo ha supuesto grandes avances en el estado del arte. En el ámbito de los juegos de estrategia destacan los agentes creados capaces de ganar los campeones mundiales de Go [2] y Dota 2 [3]. Por otro lado, en el campo de la robótica, en ramas como la manipulación de objetos, destacan los avances logrados con técnicas de aprendizaje por refuerzo aplicados sobre la mano robótica Shadow Hand [4] para el manejo preciso de objetos en el mundo real a partir del conocimiento adquirido en un entorno simulado.

No obstante, aunque ya que ha visto que los logros obtenidos mediante técnicas de aprendizaje por refuerzo se extienden a multitud de campos, este trabajo se centrará en su aplicación a la robótica móvil para, por ejemplo, permitir enseñar a los ordenadores a controlar robots en simulaciones e incluso en el mundo real.

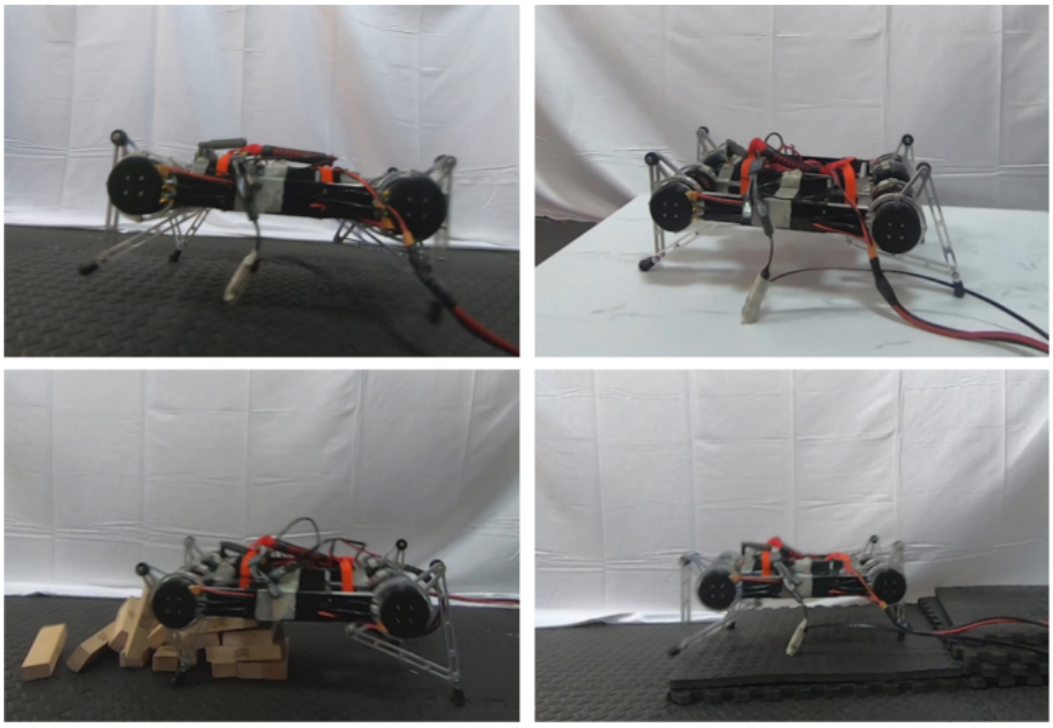
En la Figura 1 se muestran varios ejemplos de robots móviles que han aprendido a desenvolverse en escenarios simulados con características competitivas y multiagente [5]. Esto significa que en el entorno conviven más de un agente al mismo tiempo y que ambos deben de tratar de lograr un objetivo opuesto al del agente contrario.



Fuente: [5]

Figura 1: Ejemplos de agentes robóticos en entornos de simulación competitivos donde se plantean distintos escenarios (de izquierda a derecha): Correr a un objetivo, No debes pasar, Sumo y Chutar y parar.

Por otro lado, en la Figura 2 se observa un ejemplo de aplicación de aprendizaje por refuerzo empleando el robot cuadrúpedo Minitaur [4]. El objetivo buscado con este desarrollo fue presentar un método robusto de aprender el patrón de marcha directamente empleando el robot en el mundo real sin necesidad de realizar el entrenamiento en simulación.



Fuente: [4]

Figura 2: Imágenes de el patrón de marcha aprendido actuando en el mundo real. Se observa cómo con el aprendizaje obtenido, aunque el robot solo fue entrenado en terreno plano, es capaz de superar los obstáculos en su camino.

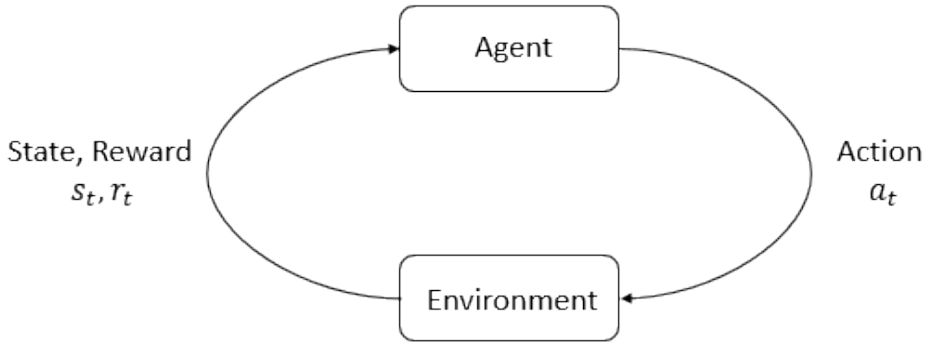
En este trabajo, se presentarán en primer lugar las bases generales de todos los algoritmos de aprendizaje por refuerzo (Sección 2), para después profundizar mencionando algunos de los algoritmos existentes. Finalmente, se reflejarán varios ejemplos de aplicación de los algoritmos de aprendizaje por refuerzo en robots móviles y se enlazarán con cuadernos en Google Colab donde probar a entrenar y evaluar el comportamiento de los agentes (Sección 3).

2. Bases del aprendizaje por refuerzo

2.1. Conceptos clave y terminología

Los dos conceptos principales del aprendizaje por refuerzo son el agente y el entorno. El entorno es el mundo en el que vive el agente y con el que interactúa. El proceso de interacción entre el entorno y el agente se divide en pasos: en el primero el agente ve una observación (posiblemente parcial) del estado del mundo y en el segundo decide qué acción tomar. De esta manera, el entorno cambia cuando el agente actúa sobre él, aunque cabe mencionar que también puede cambiar por sí solo.

El agente también percibe tras realizar cada acción una señal de recompensa del entorno, un número que le indica qué tan bueno o malo es el estado actual en el que se encuentra. El objetivo del agente es maximizar su recompensa acumulada a lo largo de todas sus acciones. Los distintos métodos o algoritmos de aprendizaje por refuerzo son formas en que el agente puede aprender comportamientos para lograr su objetivo de maximizar la recompensa acumulada. En la Figura 3 se puede observar de manera esquemática la interacción anteriormente descrita entre el agente y el entorno.



Fuente: [1]

Figura 3: Esquema de la interacción del agente con el entorno.

Para hablar más específicamente de cómo funciona el aprendizaje por refuerzo, es necesario introducir terminología adicional. Se debe hablar de

- estados y observaciones,
- espacio de acción,
- políticas,
- trayectorias,
- cálculo del retorno a partir de las recompensas,
- el objetivo del aprendizaje por refuerzo,
- y funciones valor.

2.2. Estados y observaciones

Un **estado** s es una descripción completa de cómo se encuentra el entorno. Por otro lado, una **observación** es una descripción parcial de un estado, que puede omitir información. Por ejemplo, una observación visual podría representarse mediante una imagen (una matriz RGB con unos valores para cada uno de sus píxeles) y el estado de un robot podría estar representado por los ángulos y las velocidades de cada una de sus articulaciones.

Cuando el agente es capaz de observar el estado completo del entorno, decimos que el entorno está **completamente observado**. Sin embargo, cuando el agente solo puede ver una observación parcial, decimos que el entorno se encuentra **parcialmente observado**.

2.3. Espacio de acción

Los diferentes entornos permiten diferentes tipos de acciones. El conjunto de todas las acciones válidas en un entorno dado a menudo se denomina **espacio de acción**. Algunos entornos, como Atari y Go, cuentan con un **espacio de acción discreto**, donde el agente solo dispone de un número finito de movimientos. Otros entornos, como aquellos donde el agente controla un robot en un mundo físico, tienen un **espacio de acción continuo**. En un espacio continuo, las acciones que puede tomar el agente son valores reales, no números discretos.

Esta distinción tiene algunas consecuencias bastante profundas para los métodos de aprendizaje por refuerzo. Algunas familias de algoritmos solo se pueden aplicar directamente en un caso y deben de modificarse sustancialmente para poder ser empleadas en el otro.

2.4. Políticas

La **política** es la regla que sigue el agente para decidir qué acción tomar en cada instante. Puede ser **determinista**, en cuyo caso generalmente se denota por μ:

$$a_t = \mu(s_t), \tag{1}$$

o puede ser **estocástica**, en cuyo caso generalmente se denota con π:

$$a_t \sim \pi(\cdot|s_t). \tag{2}$$

Debido a que la política es esencialmente el cerebro del agente, no es raro sustituir la palabra "política" por "agente", por ejemplo, diciendo "La política está tratando de maximizar la recompensa".

2.5. Trayectorias

Una **trayectoria** τ es una secuencia de estados y acciones en el mundo,

$$\tau = (s_0, a_0, s_1, a_1, ...). \tag{3}$$

El primer estado del entorno, s_0, se muestrea aleatoriamente a partir de la distribución que contiene todos los posibles estados iniciales, a veces denotada por ρ_0:

$$s_0 \sim \rho_0(\cdot). \tag{4}$$

Las **transiciones** de estado (lo que le sucede al entorno entre el estado en el tiempo t, s_t, y el estado en $t+1$, s_{t+1}), se rigen por las leyes del entorno y dependen solo de la acción más reciente, a_t. Las transiciones de estado pueden ser deterministas,

$$s_{t+1} = f(s_t, a_t) \tag{5}$$

o estocásticas,

$$s_{t+1} \sim P(\cdot|s_t, a_t). \tag{6}$$

Las acciones que se toman provendrán de acuerdo a la política seguida por el agente.

2.6. Recompensa y retorno

La función de **recompensa** R es de vital importancia en el aprendizaje por refuerzo. Depende del estado actual del mundo, la acción que se acaba de tomar y el próximo estado del mundo:

$$r_t = R(s_t, a_t, s_{t+1}) \tag{7}$$

aunque con frecuencia esto se simplifica a una dependencia del estado actual, $r_t = R(s_t)$, o del par estado-acción $r_t = R(s_t, a_t)$.

El objetivo del agente es maximizar la **recompensa acumulada** a lo largo de una trayectoria, a lo que se le llama **retorno**,$R(\tau)$. Sin embargo, el cálculo del retorno puede realizarse de distintas formas.

Un tipo de retorno es el **retorno sin descuento** con horizonte finito, que consiste sencillamente en calcular la suma de las recompensas obtenidas en una ventana fija de pasos (una trayectoria finita):

$$R(\tau) = \sum_{t=0}^{T} r_t. \tag{8}$$

Otro tipo de retorno es el **retorno con descuento** de horizonte infinito, que es la suma de todas las recompensas obtenidas por el agente, pero ponderada cada una de ellas con un descuento en función de cómo de alejadas en el tiempo se obtienen. Esta formulación de recompensa hace uso del factor de descuento $\gamma \in (0, 1)$:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{9}$$

La utilidad de este factor de descuento γ se puede describir tanto a nivel matemático como a nivel intuitivo. Es matemáticamente conveniente emplear un factor de descuento para ponderar las recompensas futuras pues una suma de recompensas con horizonte infinito puede no converger a un valor finito. Sin embargo, haciendo uso de un factor de descuento, la suma infinita converge. Desde un punto de vista a nivel intuitivo, sirve para modelar la incertidumbre en las recompensas que se obtendrán en el futuro, teniéndolas menos presentes conforme más alejadas en el tiempo se prevé que se logren. Por ejemplo, en términos monetarios es muy fácil de comprender: el efectivo de ahora es mejor que el efectivo futuro que una inversión pueda devolver, pues este último es más incierto.

2.7. El objetivo del aprendizaje por refuerzo

Cualquiera que sea la elección de la medida de rendimiento o del retorno (ya sea de horizonte infinito descontado o de horizonte finito sin descuento), y cualquiera que sea la política elegida, el objetivo en aprendizaje por refuerzo es seleccionar una política que maximice el rendimiento esperado cuando el agente actúa de acuerdo con ella.

Para hablar del rendimiento esperado, primero se debe hablar de distribuciones de probabilidad sobre trayectorias.

Supongamos que tanto las transiciones del entorno como la política son estocásticas. En este caso, la probabilidad de una trayectoria es:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t). \tag{10}$$

El retorno esperado (para cualquier medida), denotado por $J(\pi)$, es entonces:

$$J(\pi) = \int_{\tau} P(\tau|\pi)R(\tau) = \tau \sim \pi R(\tau). \tag{11}$$

El objetivo central que se pretende optimizar mediante aprendizaje por refuerzo puede entonces expresarse mediante

$$\pi^* = \arg\max_{\pi} J(\pi), \tag{12}$$

siendo π^* la **política óptima**.

2.8. Funciones valor

A menudo es útil conocer el valor de un estado o de un par de estado-acción. Por valor, nos referimos al rendimiento esperado si se comienza en ese estado o par estado-acción, y luego se actúa de acuerdo con una política específica. Las funciones valor se utilizan, de una forma u otra, en casi todos los algoritmos de aprendizaje por refuerzo.

Hay cuatro funciones principales a destacar:

1. ***On-Policy Value Function***, $V^{\pi}(s)$, que devuelve el retorno esperado si se comienza en el estado s y siempre se actúa de acuerdo con la política π:

$$V^{\pi}(s) = \underset{\tau\sim\pi}{E}[R(\tau)\,|s_0 = s] \tag{13}$$

2. ***On-Policy Action-Value Function***, $Q^{\pi}(s,a)$, que devuelve el retorno esperado si se comienza en el estado s, se toma una acción arbitraria a (que puede no provenir de la política), y luego se actúa indefinidamente de acuerdo con la política π:

$$Q^{\pi}(s,a) = \underset{\tau\sim\pi}{E}[R(\tau)\,|s_0 = s, a_0 = a] \tag{14}$$

3. ***Optimal Value Function***, $V^*(s)$, que devuelve el retorno esperado si se comienza en el estado s y siempre se actúa de acuerdo con la política óptima:

$$V^*(s) = \max_{\pi} \underset{\tau\sim\pi}{E}[R(\tau)\,|s_0 = s] \tag{15}$$

4. ***The Optimal Action-Value Function***, $Q^*(s,a)$, que devuelve el retorno esperado si se comienza en el estado s, se toma una acción arbitraria a, y luego se actúa indefinidamente de acuerdo con la política óptima:

$$Q^*(s,a) = \max_{\pi} \underset{\tau\sim\pi}{E}[R(\tau)\,|s_0 = s, a_0 = a] \tag{16}$$

2.9. Algoritmos empleados en aprendizaje por refuerzo

Basados en los principios anteriores, existen multitud de algoritmos de aprendizaje por refuerzo. Como ya se ha introducido anteriormente, es decisivo a la hora de elegir que

tipo del algoritmo emplear considerar si el espacio de acción es discreto o continuo. A continuación se listan algunos de los algoritmos más conocidos en función de si trabajan en espacios de acción continuos o discretos:

- Espacio de acción discreto
  - Monte Carlo
  - Q-learning
  - SARSA
  - DQN (*Deep Q-Networks*) [6]
- Espacio de acción continuo
  - DDPG (*Deep Deterministic Policy Gradient*) [7]
  - A3C (*Asynchronous Advantage Actor-Critic Algorithm*) [8]
  - NAF (*Q-Learning with Normalized Advantage Functions*) [9]
  - TRPO (*Trust Region Policy Optimization*) [10]
  - PPO (*Proximal Policy Optimization*) [11]
  - TD3 (*Twin Delayed Deep Deterministic Policy Gradient*) [12]
  - SAC (*Soft Actor-Critic*) [13]

3. Ejemplos de aplicación de algoritmos de aprendizaje por refuerzo en robots móviles

En esta sección, se pretende recoger distintos ejemplos de aplicación de algoritmos de aprendizaje por refuerzo para la resolución de tareas relacionadas con robótica móvil, específicamente con la conducción autónoma de vehículos.

Se van a presentar un conjunto de entornos de simulación para robótica móvil que pueden ser probados por cualquier usuario. Estos entornos de simulación se basan en las capacidades que proporciona *OpenAI Gym* [14], un conjunto de herramientas para desarrollar y comparar algoritmos de aprendizaje por refuerzo. *OpenAI Gym* cuenta con una gran variedad de entornos por defecto entre los cuales podemos encontrar varios enfocados en robótica móvil. Sin embargo, en este caso de estudio se va a recoger una colección de entornos desarrollada por terceros que resulta de gran interés pues se proporcionan códigos de prueba e incluso ejemplos en Google Colab para poder experimentar con ellos desde el navegador sin necesidad de instalar nada de manera local.

3.1. Entornos de simulación para la conducción autónoma

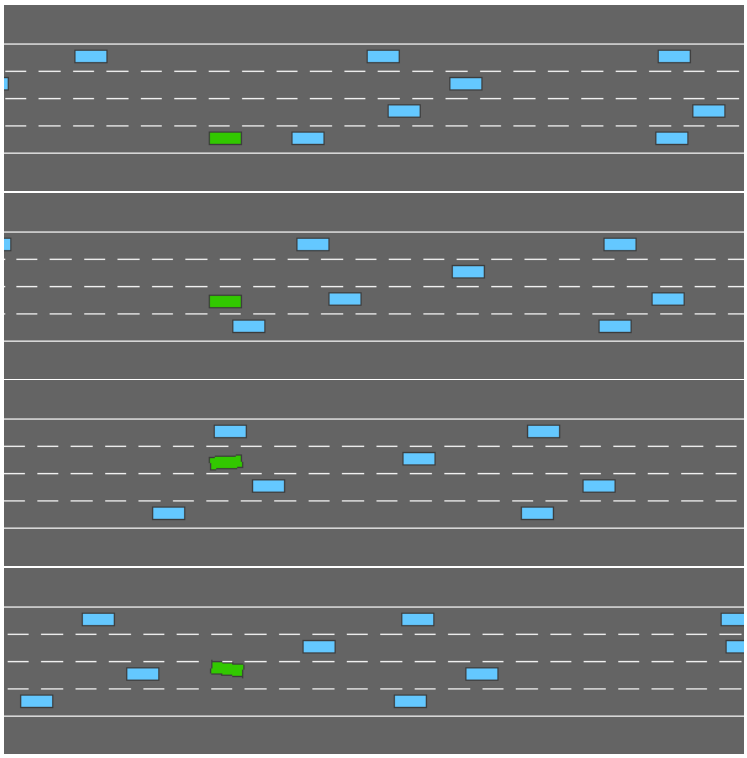
Highway-env [15] es una colección de entornos de simulación basados en *Gym* que se enfocan en la conducción autónoma. Estos entornos ofrecen la posibilidad de probar distintos algoritmos de aprendizaje por refuerzo para la toma de decisiones durante la conducción.

Como ya se ha introducido antes, estos entornos cuentan con la gran ventaja de que se proporcionan ejemplos para poder ser probados en Google Colab por cualquier usuario, facilitando la comprensión del funcionamiento de los algoritmos y permitiendo la experimentación con los distintos entornos y agentes. En la siguiente url, `https://github.com/eleurent/highway-env`, se pueden encontrar multitud de ejemplos para ser probados en Google Colab. Con estos se ejemplos, se puede entender la configuración del entorno y del agente, el proceso entrenamiento del agente y la evaluación del comportamiento del agente una vez entrenado.

A continuación se describen los entornos disponibles para ser probados.

3.1.1. Highway

En esta tarea, el vehículo debe conducir por una autovía de varios carriles ocupada por otros vehículos que circulan a distintas velocidades y que pueden ir cambiándose de carril. El objetivo del agente es alcanzar la velocidad máxima de la vía evitando colisiones con los vehículos vecinos. También se le recompensa al agente por conducir por el lado derecho de la carretera.



Fuente: [15]

Figura 4: Secuencia de imágenes del agente en el entorno "highway-v0".

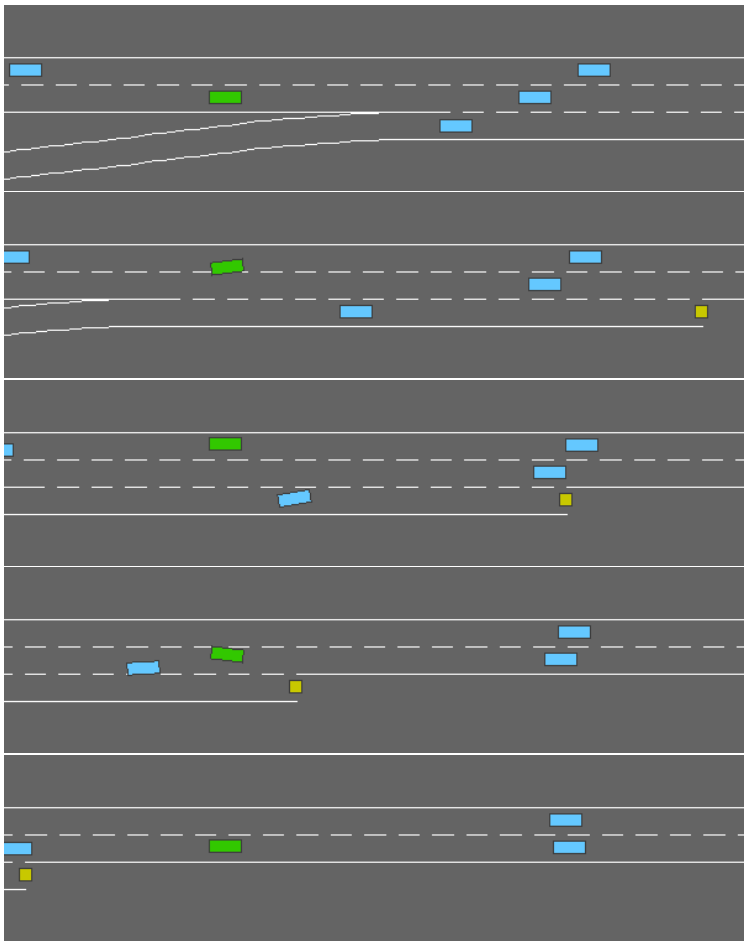
En la Figura 4 se observa una secuencia de imágenes que muestran el comportamiento aprendido por el agente. Para tratar de no reducir su velocidad, el agente realiza un adelantamiento doble y posteriormente trata de regresar al carril derecho.

En el enlace ya mencionado (`https://github.com/eleurent/highway-env`) se pueden descargar y probar en Google Colab distintas implementaciones de algoritmos de aprendizaje por refuerzo trabajando sobre este entorno. Concretamente, encontramos los siguientes dos ejemplos:

- "highway_planning.ipynb". Este agente planifica su trayectoria utilizando el algoritmo "Optimistic Planning for Deterministic Systems" (OPD) [16]. Puede probarse directamente a través del enlace `https://colab.research.google.com/github/eleurent/highway-env/blob/master/scripts/highway_planning.ipynb`.
- "sb3_highway_dqn.ipynb". En este caso el agente emplea el algoritmo Deep Q-Learning para aprender a desenvolverse en el entorno tratando de obtener la máxima recompensa posible. Este algoritmo, a diferencia de Q-Learning emplea redes neuronales para estimar la función valor Q dado un par estado-acción. Puede probarse directamente a través del enlace `https://colab.research.google.com/github/eleurent/highway-env/blob/master/scripts/sb3_highway_dqn.ipynb`.

3.1.2. Merge

En esta tarea, el vehículo comienza en una carretera principal, pero pronto se acerca a una incorporación con vehículos que pretenden incorporarse también a la vía. El objetivo del agente en este caso es mantener en la medida de lo posible la máxima velocidad de circulación de la vía mientras al mismo tiempo deja espacio para que los otros vehículos puedan incorporarse al tráfico de manera segura.



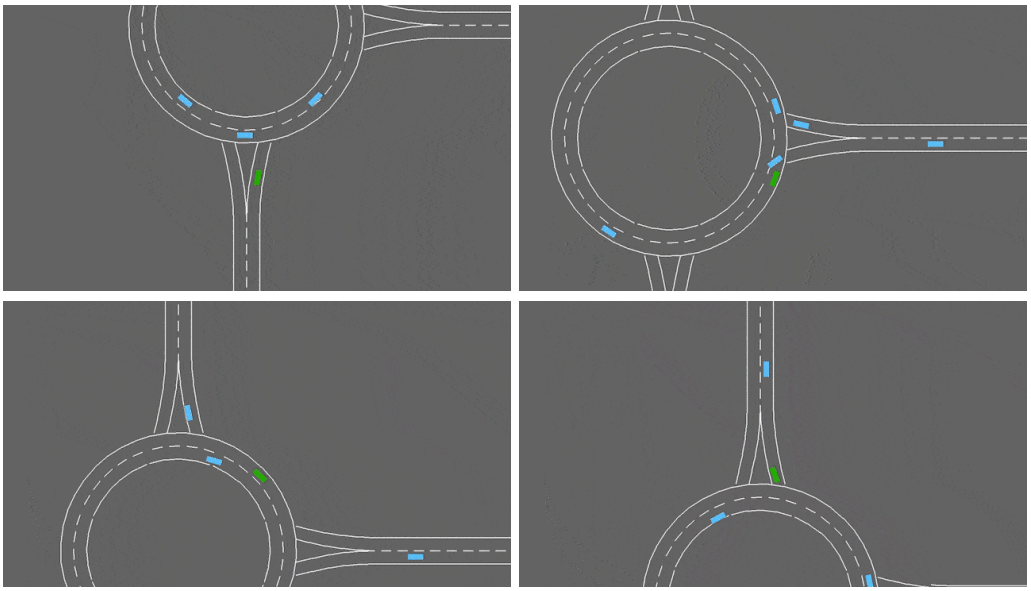
Fuente: [15]

Figura 5: Secuencia de imágenes del agente en el entorno “merge-v0”.

En la Figura 5 se observa una secuencia de imágenes que ejemplifican el comportamiento aprendido por el agente: el agente realiza un cambio al carril izquierdo para permitir la incorporación de otro vehículo y tras ello regresa al carril derecho.

3.1.3. Roundabout

En esta tarea, el vehículo se acerca a una rotonda con tráfico fluido. Seguirá la ruta planificada, que consiste en tomar la segunda salida, pero deberá lidiar con los cambios de carril, así como como con el hecho de circular por una vía no rectilínea mientras se evitan posibles colisiones con el resto de vehículos.



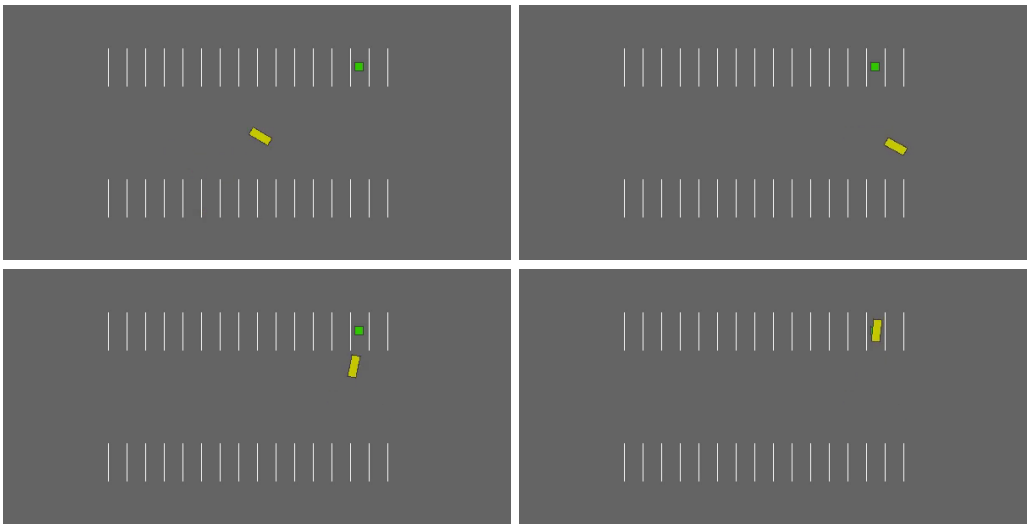
Fuente: [15]

Figura 6: Secuencia de imágenes del agente en el entorno "roundabout-v0".

En la Figura 6 se observa una secuencia de imágenes que ejemplifican el comportamiento aprendido por el agente. Tras el entrenamiento, el agente logra superar todas las dificultades que pueden surgir en este tipo de intersecciones. Antes de entrar a la rotonda, el agente ha aprendido que debe reducir su velocidad para no chocarse con los vehículos que ya circulan por la rotonda. También ha aprendido que debe reducir la velocidad en caso de que se le cruce un coche desde el carril interior de la rotonda para no chocarse con el mismo. Finalmente, el agente consigue abandonar la rotonda por la segunda salida de manera satisfactoria.

3.1.4. Parking

Se trata de una tarea de control continuo en la que el vehículo debe estacionar en una posición concreta y con una orientación específica. En la Figura 7 se puede observar la secuencia de acciones llevadas por el agente para lograr posicionarse sobre la plaza indicada dentro del aparcamiento. Para lograr la tarea, el agente ha aprendido que lo mejor es aproximarse a la posición de destino, para después maniobrar marcha atrás y estacionar.



Fuente: [15]

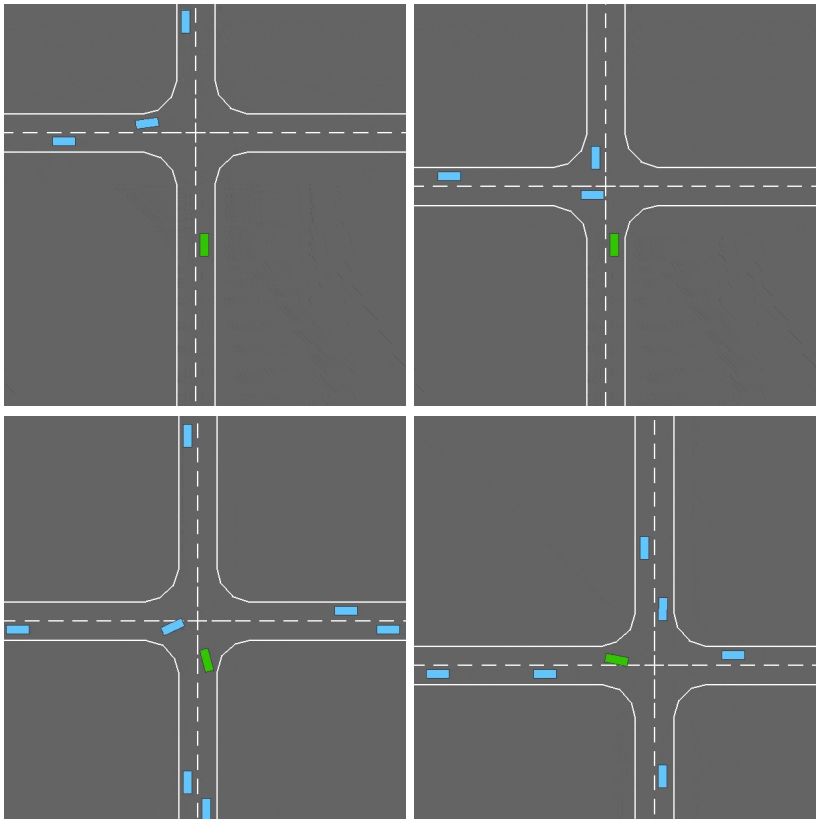
Figura 7: Secuencia de imágenes del agente en el entorno "parking-v0".

En el enlace ya mencionado (`https://github.com/eleurent/highway-env`) se pueden descargar y probar en Google Colab distintas implementaciones de algoritmos de aprendizaje por refuerzo trabajando sobre este entorno. Concretamente, encontramos las siguientes dos implementaciones:

- "parking_her.ipynb". El agente planteado en este ejemplo emplea el algoritmo Deep Deterministic Policy Gradient (DDPG) [17] para lidiar con una tarea de control continuo [7]. Hace uso también de Hindsight Experience Replay (HER) [18] para aprender de manera más eficiente cómo resolver la tarea planteada. HER, plantea la posibilidad de que el agente pueda aprender también de sus errores. En el enlace `https://colab.research.google.com/github/eleurent/highway-env/blob/master/scripts/parking_her.ipynb` se puede experimentar con el agente y el entorno anteriormente explicados.
- "parking_model_based.ipynb". En este caso se plantea un agente basado en modelo como una alternativa a todos los agentes planteados anteriormente. Resulta interesante conocer que también existen este tipo de aplicaciones, sin embargo, para implementarlas es necesario conocer al detalle el comportamiento dinámico del agente. En el enlace `https://colab.research.google.com/github/eleurent/highway-env/blob/master/scripts/parking_model_based.ipynb` se puede encontrar una explicación más detallada del planteamiento del agente así como el código necesario para probar el comportamiento del agente.

3.1.5. Intersection

En este entorno se plantea el caso de una intersección entre dos vías con carriles en ambos sentidos. El agente debe de lidiar con una tarea de tráfico denso para encontrar el momento oportuno en el que girar a la izquierda para tomar su salida.



Fuente: [15]

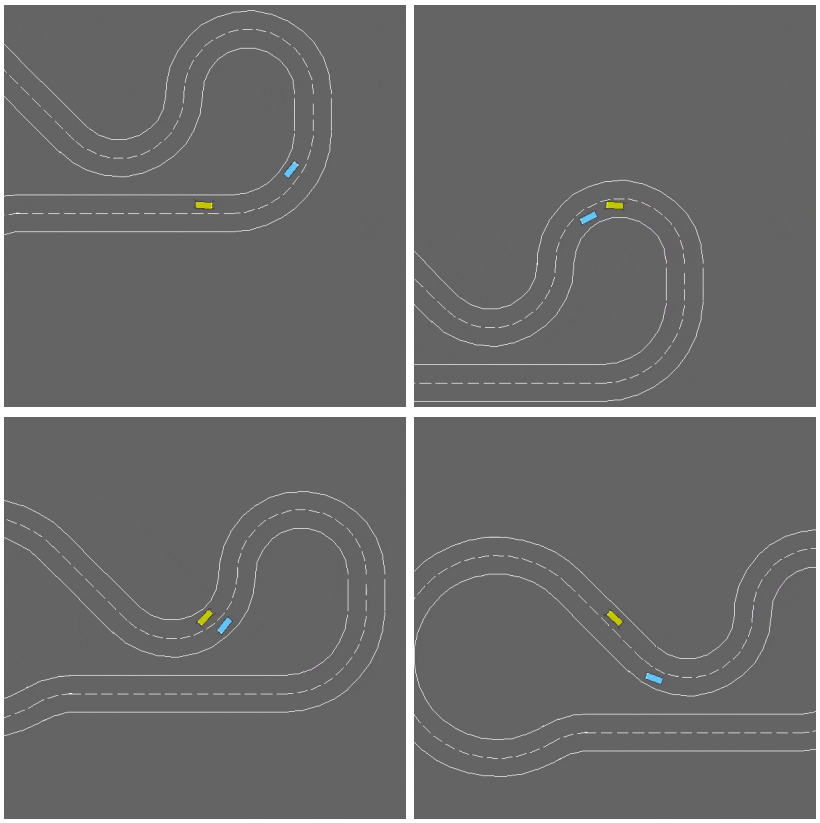
Figura 8: Secuencia de imágenes del agente en el entorno "intersection-v0".

En la Figura 8 podemos observar una serie de imágenes que muestran el resultado del proceso de aprendizaje. El agente ha aprendido que al aproximarse a la intersección debe de frenar y ceder el paso para no colisionar con los vehículos que atraviesan el cruce. Finalmente, cuando prevé que le va a dar tiempo suficiente a realizar la maniobra, realiza el giro a la izquierda de manera fluida y eficaz.

En el enlace ya mencionado (`https://github.com/eleurent/highway-env`) se puede descargar y probar en Google Colab la implementación de un algoritmo de aprendizaje por refuerzo trabajando sobre este entorno. Concretamente, encontramos la implementación siguiente: "intersection_social_dqn.ipynb" (`https://colab.research.google.com/github/eleurent/highway-env/blob/master/scripts/intersection_social_dqn.ipynb`).

3.1.6. Racetrack

Finalmente, este es el último entorno que se plantea: una carrera entre dos vehículos. Se trata de una tarea de control continuo que implica mantenerse en el carril y evitar obstáculos al mismo tiempo. El agente debe tratar de ir siempre por delante del otro vehículo y en caso de ir por detrás deberá tratar siempre de adelantarlo con seguridad.



Fuente: [15]

Figura 9: Secuencia de imágenes del agente en el entorno "racetrack-v0".

En la Figura 9 se aprecia la maniobra de adelantamiento que ha aprendido a realizar el agente (el vehículo amarillo). Se puede ver como este aprovecha el paso por curva para tener la oportunidad de adelantar al otro vehículo.

4. Conclusiones

Este trabajo refleja una alternativa a la programación tradicional en la robótica móvil. Se ha visto cómo el aprendizaje por refuerzo plantea métodos y algoritmos de aprendizaje para lograr que el agente aprenda a realizar distintas tareas a base de prueba y error. Es un paradigma que posibilita una forma de resolver los problemas de manera flexible, pues es el agente el que se encarga de adaptarse a la condiciones del entorno para tratar de maximizar la recompensa que obtiene del mismo.

Tal y como se ha visto con los ejemplos, el simulador Gym de OpenAI resulta una excelente herramienta para la simulación de entornos enfocados en el aprendizaje por refuerzo. Aunque se ha visto que existen casos donde el robot es capaz de aprender directamente en el mundo real sin necesidad de una fase previa de simulación, no es lo habitual, pues poner al robot real a realizar movimientos casi aleatorios y erráticos al principio del aprendizaje acarrea una alta peligrosidad y podría tener graves consecuencias.

Por otro lado, a través de los ejemplos enlazados a distintos cuadernos de Google Colab se pretende facilitar la comprensión del proceso de aprendizaje del agente, así como proveer una forma práctica de probar los conceptos teóricos introducidos en este trabajo.

Finalmente, en el vídeo que se enlaza a continuación se incluye una breve presentación realizada por el equipo junto con una serie de pruebas con los distintos agentes y entornos incluidos en los cuadernos de Google Colab introducidos:

`https://youtu.be/-m9PY6dYaVE`

Referencias

[1] Joshua Achiam. *Spinning Up in Deep Reinforcement Learning*. 2018.

[2] David Silver y col. "Mastering the game of Go with deep neural networks and tree search". En: *Nature* 529.7587 (ene. de 2016), págs. 484-489. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

[3] OpenAI y col. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680 [cs.LG].

[4] Tuomas Haarnoja y col. *Learning to Walk via Deep Reinforcement Learning*. 2019. arXiv: 1812.11103 [cs.LG].

[5] Trapit Bansal y col. *Emergent Complexity via Multi-Agent Competition*. 2018. arXiv: 1710.03748 [cs.AI].

[6] Volodymyr Mnih y col. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[7] Timothy P. Lillicrap y col. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG].

[8] Volodymyr Mnih y col. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. arXiv: 1602.01783 [cs.LG].

[9] Shixiang Gu y col. *Continuous Deep Q-Learning with Model-based Acceleration*. 2016. arXiv: 1603.00748 [cs.LG].

[10] John Schulman y col. *Trust Region Policy Optimization*. 2017. arXiv: 1502.05477 [cs.LG].

[11] John Schulman y col. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[12] Scott Fujimoto, Herke van Hoof y David Meger. *Addressing Function Approximation Error in Actor-Critic Methods*. 2018. arXiv: 1802.09477 [cs.AI].

[13] Tuomas Haarnoja y col. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[14] Greg Brockman y col. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[15] Edouard Leurent. *An Environment for Autonomous Driving Decision-Making*. https://github.com/eleurent/highway-env. 2018.

[16] Jean-Francois Hren y Rémi Munos. "Optimistic planning of deterministic systems". En: *European Workshop on Reinforcement Learning*. France, 2008, págs. 151-164. URL: https://hal.archives-ouvertes.fr/hal-00830182.

[17] David Silver y col. "Deterministic Policy Gradient Algorithms". En: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, 2014, I–387–I-395.

[18] Marcin Andrychowicz y col. *Hindsight Experience Replay*. 2018. arXiv: 1707.01495 [cs.LG].