



Máster en
Inteligencia
Artificial
UMU

Aprendizaje por Refuerzo

Sobre la entrega y defensa de la primera parte

Máster en Inteligencia Artificial

Universidad de Murcia

“El hombre que ha cometido un error y no lo corrige, comete otro error mayor.”

Confucio^a

^aReconocer y corregir los errores propios es una parte fundamental del crecimiento personal y el desarrollo moral. La idea central de esta frase tiene paralelismos interesantes con los algoritmos de aprendizaje por refuerzo.



15 de febrero de 2025

Índice

1. Introducción	2
2. Entregas	2
2.1. Componente Documental	3
2.2. Componente Experimental	4
2.3. Contenido del fichero .zip	5
2.4. Componente Oral	6
3. Partes del Trabajo	6
3.1. Practica 1. Bandido de k-brazos	6
3.2. Practica 2. Aprendizaje en entornos complejos	9
3.3. Investigación	10
4. Referencias para la Investigación	14
5. Uso básico de Gymnasium	18

1. Introducción

El aprendizaje por refuerzo es una de las ramas más fascinantes de la inteligencia artificial porque está demostrando que cada vez tiene más aplicaciones, desde juegos y finanzas en sus inicios hasta la construcción LLMs (Large Language Models) como DeepSeek. Su enfoque se basa en la interacción de un agente con un entorno para aprender a tomar decisiones óptimas a través de la experiencia y la retroalimentación.

El objetivo de su trabajo final es que desarrollen las actividades teórico-prácticas que se proponen para que profundicen en algunos conceptos y fundamentos clave del aprendizaje por refuerzo. Pero más allá del desarrollo de los ejercicios que se proponen, en un máster es importante la búsqueda y selección de bibliografía relevante. Es fundamental que consulten fuentes confiables, como artículos científicos, libros y recursos académicos, y que los integren adecuadamente en sus análisis.

Con todo ello, tal y como se indica en la guía docente, se trata de que entreguen un trabajo donde muestren creatividad, precisión, rigor técnico y profundidad de análisis, discusión de resultados, justificación de las decisiones adoptadas, dando conclusiones razonadas y avaladas por fuentes acordes al informe realizado.

A continuación se indican las actividades teórico-prácticas que deben desarrollar así como varias alternativas de trabajos que pueden exponer y presentar como complemento a la actividades anteriores.

2. Entregas

Deberán de desarrollar **un trabajo** con consta de 3 partes:

- ▶ Práctica 1: Problema del Bandido de k-brazos.
- ▶ Práctica 2: Técnicas Básicas de Aprendizaje por Refuerzo.
- ▶ Investigación.

Cada parte constará de dos componentes:

- ▶ la documental, que será un fichero . pdf que fundamente la componente experimental, y
- ▶ la experimental, que podrá ser ejecutada en Colab (**sin errores**) a partir de un repositorio de GitHub y que justifique la componente documental.

Además, para la parte de investigación, se tendrá una componente oral.

Para tener una evidencia completa, a efectos de evaluación y custodia, en el Aula Virtual tendrán que presentar todas las partes con todas sus componentes en **un único fichero .zip**. En concreto, para la parte experimental, se deberá presentar la versión del repositorio con la que hayan realizado la parte documental. La entrega de la componente experimental en el Aula Virtual no exime de la exigencia de que se pueda ejecutar en Colab. Para la evidencia de la componente oral de la parte de investigación será grabada por videoconferencia (zoom).

! Importante

- ▶ Las exposiciones tendrán lugar el 3 y el 6 de marzo. Las horas de esos días se dedicarán en exclusividad a las exposiciones, sin importar el número de presentados.
- ▶ La entrega de las tres partes, **en un único fichero .zip**, tendrá como fecha tope el 9 de marzo, y se hará por el Aula Virtual (herramienta Tareas).

2.1. Componente Documental

Para la parte documental de cada parte es importante que estructuren las entregas de manera clara, incluyendo:

1. **Título y Autores:** Incluir el nombre de la obra y los nombres completos de los autores del trabajo, junto con el correo electrónico um.es para cada autor.
2. **Introducción:**
 - ▶ Breve descripción del problema y su relevancia.
 - ▶ Motivación del trabajo: ¿Por qué es importante el problema estudiado?
 - ▶ Objetivos del informe: ¿Qué se pretende lograr con el desarrollo del trabajo?
 - ▶ Organización del documento: breve explicación de la estructura del informe.
3. **Desarrollo:**
 - ▶ Definición formal del problema o aplicación específica abordada.
 - ▶ Describir enfoques existentes en la literatura sobre problemas similares.
 - ▶ Contexto y antecedentes necesarios para entender el trabajo.
 - ▶ Explicación de los métodos utilizados para abordar el problema.
 - ▶ Justificar cómo el presente trabajo se diferencia o mejora enfoques previos.
4. **Algoritmos:**
 - ▶ Explicar en detalle los pasos de los algoritmos empleados.
 - ▶ Incluir pseudocódigo y, opcionalmente, diagramas de flujo si es necesario.
 - ▶ Justificar la elección de los algoritmos y su aplicabilidad al problema estudiado.
5. **Evaluación/Experimentos:**
 - ▶ Configuración experimental: herramientas, entornos de prueba, datasets empleados, etc.
 - ▶ Métricas utilizadas para evaluar el desempeño del método propuesto.
 - ▶ Resultados obtenidos, presentados en tablas o gráficos según corresponda.
 - ▶ Análisis crítico de los resultados y comparación con otros enfoques.
6. **Conclusiones:**
 - ▶ Un resumen de los principales resultados obtenidos.
 - ▶ Limitaciones del estudio y posibles mejoras futuras.
 - ▶ Reflexión sobre la importancia del trabajo y su impacto en el campo del aprendizaje por refuerzo.
 - ▶ Líneas *futuras* de estudio.
7. **Bibliografía:** Listar todas las referencias utilizadas en el trabajo. Se recomienda utilizar un gestor de referencias como BibTeX para facilitar la organización de las citas.

! Importante

- ▶ Incluir en las referencias el enlace URL donde se encuentre el repositorio GitHub donde reproducir lo documentado.
- ▶ Todos los documentos se desarrollarán **de acuerdo a la plantilla** que tienen en recursos del Aula Virtual.

Cada apartado podrá contener subapartados pero cada apartado del documento se desarrollará con la extensión que sea acorde a lo que se presenta.

Por ejemplo, en el apartado de algoritmos, si es respecto a una práctica y utiliza tres (por poner una cantidad) bastará con mencionarlos (pues ya se ha explicado en clase) salvo que quiera destacar alguna particularidad sobre ello. Sin embargo, en el apartado de algoritmos, si es respecto a la parte de investigación, lo que se espera es una explicación más detallada y justificada.

Para la práctica 1 y la práctica 2 deberán desarrollar documentos muy breves (posiblemente con 2 o 3 páginas sea suficiente). Se espera que, para las prácticas, los .pdf sean un resumen escueto de lo que presentan en los notebooks. Los notebooks podrán tener toda la extensión que consideren.

Por contra, para el trabajo de investigación, se espera un documento más extenso y aunque el alcance del trabajo lo establecerá el grupo deberá estar entre 5 y 10 páginas.

En cualquier caso **sean claros y concisos** en su presentación, enfocándose en los **puntos clave** sin omitir de mencionar y explicar los **detalles importantes**.

2.2. Componente Experimental

Cada parte (Práctica 1, Práctica 2 e Investigación) tendrá su componente experimental que respaldará la componente documental. Cada componente experimental constará de tantos notebooks y ficheros .py como considere necesarios, siempre que sirvan de respaldo a lo documentado.

Cada parte tendrá su correspondiente respaldo en **GitHub** y su contenido debe seguir las siguientes instrucciones:

1. Crear un repositorio llamado **trabajo_NombreGrupo**, donde **NombreGrupo** es el nombre del grupo. Por ejemplo, las iniciales de los apellidos de sus miembros. El nombre no tendrá más de 8 caracteres.
2. Incluir los *notebooks* **ya resueltos** en el repositorio.
3. Incluir en el repositorio un fichero README.md con los siguientes datos.

```
# Título del Trabajo

## Información
- **Alumnos:** Apellido1, Nombre1; Apellido2, Nombre2; ...
- **Asignatura:** Nombre de la asignatura
- **Curso:** 2024/2025
- **Grupo:** NombreGrupo

## Descripción
[Breve descripción del trabajo y sus objetivos]

## Estructura
[Explicación de la organización del repositorio]


## Instalación y Uso
[Instrucciones si son necesarias]

## Tecnologías Utilizadas
[Lista de lenguajes, frameworks, etc.]
```

4. El repositorio tendrá la siguiente estructura general:

```
trabajo_NombreGrupo/
|__ src/          # Código .py (expanda src, o considere varios src con distintos nombres)
|__ docs/         # Documentación ( .bib, .pdf, ... si procede)
|__ tests/        # Tests (si aplica)
|__ data/         # Datos necesarios (si aplica)
|__ README.md
|__ main.ipynb    # Fichero principal
|__ notebook1.ipynb # Breve introducción del problema y enlace a los estudios.
|__ notebook2.ipynb # Estudio de la familia epsilon-greedy
|__ etc.ipynb     # Poner tantos como estudios realizados (al menos 5)
```

5. Todo se podrá ejecutar en Colab.

El fichero **main.ipynb** comenzará con el típico enlace de  **Open in Colab**, que abrirá el notebook en Colab. https://colab.research.google.com/github/autores/trabajo_NombreGrupo/blob/main/main.ipynb
Las primeras celdas del *notebook* principal harán una copia del repositorio y se instalarán los paquetes que sean necesarios. A partir de dicho *notebook* se podrá navegar entre los distintos *notebooks*.

! Importante

- ▶ El enlace se consigue de forma automática al salvar el *notebook* desde Colab en el repositorio.
- ▶ Ya que se tiene tres partes, se tendrán **tres repositorios**. Sus nombres serán: *investigacionRL_NombreGrupo*, *k_brazos_NombreGrupo* y *RL_NombreGrupo*.
- ▶ Asegúrese de que cada *notebook* tenga la **semilla** de generación de números aleatorios que permita reproducir las gráficas o tablas de las componentes documentadas.
- ▶ El **prerequisito** para la evaluación es que al entrar por primera vez en el *notebook* y seleccionar la opción Entorno de ejecución > Ejecutar todas **no se produzcan errores** y el resultado final sea exactamente el mismo al que se encuentra en el repositorio. Es decir, que sus notebooks **son reproducibles**.
 - Tendrán que garantizar que las celdas iniciales de cada *notebook* realicen todo lo necesario para que no se produzca error alguno en su ejecución.
 - La no-ejecución completa de un notebook, conlleva que todo su contenido quedará descartado y no se considerará superado.

2.3. Contenido del fichero .zip

En el Aula Virtual deberá de presentar la evidencia del trabajo realizado en **un único fichero .zip** que contenga un fichero .pdf por cada parte junto con la versión del repositorio que respalda el documento. De acuerdo a lo explicado anteriormente, se espera que en la descompresión del fichero se muestre la estructura general de la Figura 1.

```
NombreGrupo
|-- investigacionRL_NombreGrupo
|   |-- __investigacionRL_NombreGrupo (github)
|   |   |-- README.md
|   |   |-- data
|   |   |-- docs
|   |   |-- etc.ipynng
|   |   |-- main.ipynng
|   |   |-- notebook1.ipynng
|   |   |-- notebook2.ipynng
|   |   |-- src
|   |   |-- tests
|   |-- __investigacionRL_NombreGrupo.pdf
|-- k_brazos_NombreGrupo
|   |-- __k_brazos_NombreGrupo (github)
|   |   |-- README.md
|   |   |-- data
|   |   |-- docs
|   |   |-- etc.ipynng
|   |   |-- main.ipynng
|   |   |-- notebook1.ipynng
|   |   |-- notebook2.ipynng
|   |   |-- src
|   |   |-- tests
|   |-- __k_brazos_NombreGrupo.pdf
|-- RL_NombreGrupo
|   |-- __RL_NombreGrupo (github)
|   |   |-- README.md
|   |   |-- data
|   |   |-- docs
|   |   |-- etc.ipynng
|   |   |-- main.ipynng
|   |   |-- notebook1.ipynng
|   |   |-- notebook2.ipynng
|   |   |-- src
|   |   |-- tests
|   |-- __rl_NombreGrupo.pdf
```

Figura 1: Estructura esperada del fichero .zip

! Importante

Es fundamental que todo el fichero .zip se descomprima **a partir de un directorio** cuyo nombre se identifique con el nombre del grupo.

2.4. Componente Oral

Solo los que vayan a presentar las 3 partes del trabajo podrán realizar la exposición de la parte de investigación.

De acuerdo al número de alumnos matriculados, si todos se presentaran a la exposición por videoconferencia (zoom), cada uno tendrá una intervención de unos 4', supuesto que no hay descansos y no hay problemas técnicos.

! Importante

A efectos de conocer el tiempo real de exposición que tendrá cada grupo, deberán notificar, a más tardar el **28 de febrero** y antes de las 14h, quiénes presentarán el trabajo completo. El mismo día 28 se anunciará el orden de defensa de los trabajos, que se seleccionarán de forma aleatoria.

3. Partes del Trabajo

Como se ha indicado antes, el trabajo consta de 3 partes: práctica 1, práctica 2 e investigación. A continuación se describe cada una de ellas.

3.1. Practica 1. Bandido de k-brazos

En esta parte se analizan los algoritmos del bandido de k -brazos, comparando su rendimiento en entornos estacionarios. Para ello se implementan varias familias de métodos: ϵ -greedy, UCB, ascenso del gradiente, evaluando las recompensas y rechazos acumulados.

El **problema del bandido de k -brazos** modela una situación de toma de decisiones secuencial bajo incertidumbre. Consiste en lo siguiente:

- ▶ Una máquina tragaperras con k brazos (acciones)
- ▶ Cada brazo i tiene una distribución de recompensa desconocida \mathcal{P}_i
- ▶ En cada paso t , el agente:
 1. Elige un brazo $a_t \in \{1, \dots, k\}$
 2. Recibe una recompensa $r_t \sim \mathcal{P}_{a_t}$
- ▶ El objetivo es maximizar la **recompensa acumulada** $\sum_{t=1}^T r_t$ donde T es el horizonte temporal.
- ▶ Una política es tomar decisiones que tengan en cuenta aquellos brazos que, por la experiencia, han mostrado tener mayores recompensas.
- ▶ Alternativamente, otra forma de medir las mejores decisiones es considerar como medida de desempeño el **rechazo** (*regret*) que cuantifica la pérdida por no elegir siempre el mejor brazo:

$$R(T) = T\mu^* - \mathbb{E} \left[\sum_{t=1}^T \mu_{a_t} \right]$$

donde $\mu^* = \max_i \mu_i$ y $\mu_i = \mathbb{E}[\mathcal{P}_i]$.

- ▶ La consideración del rechazo da lugar al estudio de una multitud de algoritmos alternativos.

El problema del bandido multibrazo es el mejor ejemplo del dilema de la **Exploración-Explotación**. Los mejores algoritmos son los que son capaces de encontrar el mejor equilibrio entre ambas opciones.

Las tareas a realizar en esta primera parte son dos.

- Completar el estudio de la familia ϵ -greedy
- Implementación y estudio de otras familias

A) Completar el estudio de la familia ϵ -greedy

Usando https://github.com/ldaniel-hm/eml_k_bandit/blob/main/bandit_experiment.ipynb trabajaremos con el repositorio https://github.com/ldaniel-hm/eml_k_bandit.

Es un ejemplo de cómo se pueden estudiar los algoritmos del bandido de k-brazos. En este caso se comparan tres algoritmos ϵ -greedy. El código del repositorio y del notebook inicial están incompletos: queremos hacer un estudio más completo sobre este conjunto de técnicas.

Añade más gráficos. Se consideran *necesarias* las siguientes gráficas:

- Mostrar el porcentaje en el que fue seleccionado el grafo óptimo.

```
def plot_optimal_selections(steps: int,
                           optimal_selections: np.ndarray,
                           algorithms: List[Algorithm]):
    """
    Genera la gráfica de Porcentaje de Selección del Brazo Óptimo vs Pasos de Tiempo.

    :param steps: Número de pasos de tiempo.
    :param optimal_selections: Matriz de porcentaje de selecciones óptimas.
    :param algorithms: Lista de instancias de algoritmos comparados.
    """
```

- Mostrar las estadísticas de cada brazo. Cada valor en el eje X representará un brazo. En el eje Y se representa el promedio de las ganancias de cada brazo. El gráfico mostrará un histograma para cada brazo donde se muestre el promedio de las ganancias obtenidas pero en el eje X, como etiqueta, se mostrará también el número de veces que fue seleccionado el brazo e indicar si es el brazo óptimo o no.

Añade lo que considere necesario que ayude a clarificar el significado de la gráfica.

```
def plot_arm_statistics(arm_stats: LoQueConsideres,
                       algorithms: List[Algorithm], *args):
    """
    Genera gráficas separadas de Selección de Arms:
    Ganancias vs Pérdidas para cada algoritmo.

    :param arm_stats: Lista (de diccionarios) con estadísticas de cada brazo por algoritmo.
    :param algorithms: Lista de instancias de algoritmos comparados.
    :param args: Opcional. Parámetros que consideres
    """
```

- Mostrar la evolución del rechazo, de forma análoga a cómo se muestra la evolución de las recompensas.

```
def plot_regret(steps: int,
                regret_accumulated: np.ndarray,
                algorithms: List[Algorithm], *args):
    """
    Genera la gráfica de Regret Acumulado vs Pasos de Tiempo

    :param steps: Número de pasos de tiempo.
    :param regret_accumulated: Matriz de regret acumulado (algoritmos x pasos).
    :param algorithms: Lista de instancias de algoritmos comparados.
    :param args: Opcional. Parámetros que consideres. P.e. la cota teórica  $Cte * \ln(T)$ .
    """
```


Qué debe de hacer

- ▶ Modifique los ficheros https://github.com/ldaniel-hm/eml_k_bandit/blob/main/bandit_experiment.ipynb y https://github.com/ldaniel-hm/eml_k_bandit/blob/main/plotting/plotting.py para incluir las gráficas indicadas.

Se pueden modificar los parámetros de las funciones de dibujo si se viera conveniente (aquí solo se da una propuesta) mientras que cumpla con su objetivo. Obviamente, también se modificará el resto del código si se viera necesario.

- ▶ Complete el estudio de la familia de algoritmos ϵ -greedy e indique qué gráficas son las más relevantes ¿por qué?

B) Implementación y estudio de otras familias

Con la siguiente propuesta se trata de hacer estudios comparativos entre métodos de la misma familia. En concreto.

- ▶ Métodos UCB: UCB1 y UCB2.
- ▶ Métodos de Ascenso del Gradiente: Softmax y Gradiente de Preferencias.

Eso sí, tenga en cuenta que no tiene por qué mostrar los 4 gráficos del primer estudio (apartado anterior), puede que no le aporte nada tener tantos gráficos. Por eso se pide en el apartado anterior cuál de ellos son los importantes. Use solo los relevantes.

Contemple el estudio usando los siguiente tipos de bandidos.

- ▶ Distribución Bernoulli. Cada brazo solo puede dar dos resultados: éxito, 1, o fracaso, 0.

Ejemplo: Un usuario hace clic o no en un anuncio publicitario (Click-Through Rate).

- ▶ Distribución Binomial, $\mathcal{B}(n, p)$. Es cuando nos referimos a que, si lo tiramos n veces, el **número total de éxitos** (recompensas positivas) se dará con probabilidad p . La distribución Bernoulli es un caso particular de la binomial, para $n = 1$.

Ejemplo: Publicidad Online. Supongamos que un anunciante quiere optimizar su campaña de publicidad en línea eligiendo entre $k = 3$ diferentes anuncios. Cada anuncio tiene una probabilidad desconocida de que un usuario haga clic en él (CTR - Click-Through Rate). A_1 tiene un CTR desconocido de p_1 , A_2 tiene un CTR desconocido de p_2 y A_3 tiene un CTR desconocido de p_3 .

El anunciante usa un **modelo de bandido de k -brazos**, donde cada vez que muestra un anuncio (elige un brazo), observa si el usuario hizo clic (éxito = 1) o no (fracaso = 0). Después de mostrar (explorar) el anuncio A_j un total de n_j veces y recibir (aprender) X_j clics, el número de éxitos sigue una distribución binomial: $X_j \sim \text{Bin}(n_j, p_j)$. Entoces se mostrará (explotar) el anuncio que tiene mayor p_j .

- ▶ Distribución Normal, $\mathcal{N}(\mu, \sigma^2)$. Si $X \sim \mathcal{B}(n, p)$ con $np \geq 5$ y $n(1 - p) \geq 5$ se garantiza que la distribución binomial no está demasiado sesgada y tenga una forma aproximadamente simétrica. Entonces se puede hacer una **aproximación normal**: $X \approx \mathcal{N}(\mu, \sigma^2)$ donde: $\mu = np$ y $\sigma^2 = np(1 - p)$.

! Importante

- ▶ **Para cada estudio** construya un notebook diferente.
- ▶ Utilice siempre la misma semilla para la generación de números aleatorios para que sus gráficas sean reproducibles.

3.2. Practica 2. Aprendizaje en entornos complejos

El bandido de k -brazos permite estudiar el problema de la explotación y exploración en entornos que no cambian. Sin embargo, lo cierto es que cada toma de decisiones se desarrolla en un contexto. Una forma de abordar el problema es mediante bandidos contextuales. Otra forma, la más habitual, es considerar que las decisiones se rigen por un proceso de decisión de Markov con recompensas (que podrían ser estocásticas). La forma más correcta de resolver este problema es mediante las ecuaciones de Bellman que se pueden resolver mediante cálculo matricial o cálculo iterativo, para lo que se supone que conoce todo (la política correcta, la matriz de transiciones y las recompensas).

Pero lo cierto es que en este tipo de problemas no se suele conocer el modelo que rige el entorno ni se sabe que recompensas nos podemos encontrar y mucho menos cuál es la mejor decisión que se debería de tomar en un estado. Para solucionar el problema se recurre a la experiencia del agente cuando transita por el entorno. Así surge un grupo de técnicas que se basan en aplicar la siguiente secuencia de pasos:

1. El agente observa el estado actual del entorno.
2. Basándose en este estado, el agente toma una acción (de acuerdo a una política) si no está en un estado terminal.
3. El entorno cambia a un nuevo estado como resultado de la acción.
4. El agente recibe una recompensa que se usará para evaluar la acción tomada.
5. Se vuelve al paso 1.

Las técnicas de Monte Carlo esperan a que el agente llegue al estado terminal para considerar el episodio y a partir de él actualiza su política (la toma de decisiones en cada estado) para maximizar la recompensa acumulativa futura. Las técnicas de Diferencias Temporales van actualizando la política conforme va tomando decisiones, sin esperar a llegar a un estado terminal. En medio están las que consideran solo algunas primeras recompensas del episodio. Estas técnicas se basan en usar una representación tabular de las funciones de evaluación de estados y acciones.

En ocasiones la cantidad de estados y acciones es tan enorme, que no queda más remedio que recurrir a funciones aproximadas de las funciones de evaluación, lo que da lugar a todo un conjunto de técnicas basadas en el método del descenso del gradiente (como el uso de redes neuronales). Alternativamente, otro grupo de técnicas se enfocan en optimizar la política directamente lo que da lugar a los métodos Actor-Crítico.

Las tareas a realizar en esta parte son las siguientes.

- ▶ Conocer el entorno Gymnasium
- ▶ Estudio de algunos algoritmos básicos

A) Conocer el entorno Gymnasium

Gymnasium (<https://gymnasium.farama.org/>) es un entorno que permite ensayar con distintos algoritmos de aprendizaje. En clase se explicó tanto la generación de episodios como el diseño de agentes de aprendizaje. Como ejemplo de uso se comentó el contenido de https://github.com/ldaniel-hm/eml_tabular/blob/main/MonteCarloTodasLasVisitas.ipynb%20-%20Colab.pdf haciendo hincapié en la versión de Monte Carlo implementada, donde se calcula *hacia adelante* en vez de *hacia atrás* y por qué en este caso $f(t) = \frac{\sum_{i=1}^t R_i}{t}$ es un indicador de una evolución correcta del algoritmo de aprendizaje.

Antes de continuar conviene que lea el último apartado (sección 5).

Estudie el código de

https://github.com/ldaniel-hm/eml_tabular/blob/main/MonteCarloTodasLasVisitas.ipynb

y **modifique el código** para que haga lo siguiente:

- ▶ Construya un agente de acuerdo a las indicaciones de Gymnasium para que aprenda de acuerdo al algoritmo explicado en el notebook. El diseño le servirá para usarlo en el resto de la práctica.
- ▶ Añada una gráfica más. Construya la gráfica de $f(t) = \text{len}(\text{episodio}_t)$. ¿Por qué esta gráfica también es un buen indicador de aprendizaje?

B) Estudio de algunos algoritmos básicos

Desarrollará estudios comparativos entre las distintas técnicas vistas en clase.

- ▶ **Métodos Tabulares:** Monte Carlo (on-policy y off-policy), Diferencias temporales (SARSA, Q-Learning)
- ▶ **Control con Aproximaciones:** SARSA semi-gradiente, Deep Q-Learning.

Construya los correspondientes agentes de aprendizaje y seleccione al menos dos entornos de Gymnasium (<https://gymnasium.farama.org/>) donde uno de ellos ponga de manifiesto la necesidad de usar métodos aproximados.

Notar que no se le pide ningún algoritmo Actor-Crítico (si quiere implementar alguno se hará en la parte de investigación).

! Importante

Use la representación gráfica o tabla de valores que considere más conveniente para mostrar el aprendizaje de los agentes. Intente usar representaciones o tablas acordes al tipo de aprendizaje.

3.3. Investigación

A continuación se presentan varias líneas de trabajo. Tienen que seleccionar una; pero tengan en cuenta que algunas de ellas les puede costar más trabajo si aún no hemos dado sus correspondiente fundamentos teóricos.

El trabajo elegido será el que tengan que exponer al resto de compañero. Se trata de considerar alguna(s) técnica(s) novedosa(s) y presenta un estudio sobre ella(s) y, a ser posible, compararla(s) con las desarrolladas en alguna de las prácticas. Por ejemplo si eligiese realizar un trabajo sobre el problema del bandido de k-brazos podría presentar los algoritmos LinUCB y ϵ -Greedy para Bandidos Contextuales y compararlos con el algoritmo UCB1.

Las propuestas de investigación son las siguientes.

- Bandidos Duelistas

Los *bandidos duelistas* son una extensión del problema clásico de los *bandidos multi-brazo*. En lugar de recibir una recompensa numérica fija por cada acción, el agente selecciona dos acciones y obtiene información en forma de comparación (*duelo*) sobre cuál es mejor. Este enfoque es particularmente útil en escenarios donde las recompensas absolutas no están disponibles o son difíciles de definir. El enfoque intuitivo es que los humanos rara vez asignan una puntuación exacta a una opción, pero sí pueden decir cuál prefieren entre dos alternativas. El modelo de bandidos duelistas se asemeja más a cómo tomamos decisiones en la vida cotidiana.

Encontramos una aplicación en sistemas de recomendación (como Netflix o Spotify), es difícil cuantificar la satisfacción del usuario con una única métrica. Comparar dos recomendaciones en términos de preferencia

relativa es una solución más efectiva. También tiene aplicación en motores de búsqueda, donde los algoritmos pueden evaluar qué combinación de resultados ofrece una mejor experiencia al usuario sin depender de una métrica fija como los clics. Esto mismo lo habrá podido comprobar en ChatGPT que, de vez en cuando, le ofrece dos posibles respuestas y le pide que seleccione una de ellas.

- Bandidos Adversarios

*"Un bandido estocástico con K acciones está completamente determinado por las distribuciones de recompensas, P_1, \dots, P_K , de las respectivas acciones. En particular, en la ronda t , la distribución de la recompensa X_t recibida por un aprendiz que elige la acción $A_t \in [K]$ es P_{A_t} , independientemente de las recompensas y acciones pasadas. Si se observa detenidamente cualquier problema del mundo real, ya sea el diseño de medicamentos, la recomendación de artículos en la web o cualquier otro, pronto descubriremos que en realidad no existen distribuciones adecuadas. En primer lugar, será difícil argumentar que la recompensa se genera realmente de manera aleatoria y, aun si fuera generada aleatoriamente, las recompensas podrían estar correlacionadas en el tiempo. Tener en cuenta todos estos efectos haría que un modelo estocástico sea potencialmente bastante complicado. Una alternativa es adoptar un enfoque pragmático, en el que casi nada se asuma sobre el mecanismo que genera las recompensas, pero manteniendo el objetivo de competir con la mejor acción en retrospectiva. Esto nos lleva al llamado **modelo de bandido adversarial**" <https://banditalgs.com/2016/10/01/adversarial-bandits/>.*

El problema del *bandido adversarial* es una variante del problema del bandido multibrazo en la que las recompensas asociadas a cada acción pueden ser determinadas por un adversario, en lugar de seguir una distribución fija y desconocida. Este enfoque es relevante en escenarios donde las recompensas pueden cambiar de manera adversa o estratégica

- Bandidos Contextuales

En la mayoría de los problemas de bandidos, es probable que haya alguna información adicional disponible al comienzo de las rondas y, a menudo, esta información puede ayudar potencialmente con las opciones de acción. Por ejemplo, en un sistema de recomendación de artículos web, donde el objetivo es mantener a los visitantes interesados en el sitio web, la información contextual sobre el visitante del sitio web, la hora del día, información sobre lo que está de moda, etc., probablemente puede mejorar la elección del artículo que se colocará en la "portada". Por ejemplo, es más probable que un artículo orientado a la ciencia capte la atención de un fanático de la ciencia, y es posible que a un fanático del béisbol le importe poco el fútbol europeo.

Si utilizáramos un algoritmo estándar de búsqueda de artículos (como UCB), el enfoque único que adoptan estos algoritmos, que solo buscan encontrar el artículo más atractivo, probablemente decepcionará a una parte innecesariamente grande de los visitantes del sitio. En situaciones como esta, dado que el parámetro de referencia que los algoritmos de búsqueda de artículos buscan alcanzar funciona mal al omitir la información disponible, es mejor cambiar el problema y redefinir el parámetro de referencia. Sin embargo, es importante darse cuenta de que aquí hay una dificultad inherente. <https://banditalgs.com/2016/10/14/exp4/>

El problema del *bandido contextual* es una extensión del problema del bandido multibrazo que incorpora información contextual en el proceso de toma de decisiones. En cada ronda, el agente observa un contexto y debe seleccionar una acción (o brazo) basándose en este contexto, con el objetivo de maximizar la recompensa acumulada a lo largo del tiempo. Este enfoque es especialmente relevante en aplicaciones como sistemas de recomendación, publicidad en línea y personalización de contenido.

- Planificación.

En la práctica no tenemos ningún modelo del entorno y el agente debe aprender directamente de la experiencia, interactuando con el entorno. Esto da lugar a algoritmos como Q-Learning, SARSA, etc ... Pero podríamos usar la experiencia para construir un modelo. Si cada vez que se da un paso almaceno la

muestra $(S_t, A_t, R_{t+1}, S_{t+1})$ podré tener estimaciones de $p(s', r|s, a)$. Entonces tendría lotes de muestras que podría utilizar una y otra vez para hacer estimaciones de los valores de los estados y las acciones. A esto se le llama planificación. Está comprobado que la planificación mejora cualquier método (de hecho se usa en DQLearning). Hay varias formas de hacerlo ¿en cuanto mejora la planificación la resolución de las tareas en los entornos que has seleccionado en la segunda práctica?

- TD(λ)

El algoritmo TD(λ) es una técnica de *Aprendizaje por Refuerzo* que combina los métodos de *Diferencia Temporal* (TD) y *Monte Carlo*, utilizando un parámetro de trazas de elegibilidad, λ , para equilibrar entre ambos enfoques. Fue introducido por Richard S. Sutton en 1988 y se ha convertido en una herramienta fundamental en el aprendizaje automático.

La actualización del valor de un estado s_t en TD(λ) se define como: $V(s_t) \leftarrow V(s_t) + \alpha \delta_t e_t$ donde

- ▶ α es la tasa de aprendizaje,
- ▶ δ_t es el error de TD, calculado como: $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$
- ▶ e_t es la traza de elegibilidad, que se actualiza según: $e_t = \gamma \lambda e_{t-1} + 1$

Aquí, γ es el factor de descuento y λ es el parámetro que controla la longitud de las trazas de elegibilidad.

TD(λ) es un método de aproximación porque no calcula los valores exactos de los estados, sino que estima la función de valor utilizando las trazas de elegibilidad para interpolar entre los métodos TD(0) y Monte Carlo. Ajustando el parámetro λ , se puede controlar el equilibrio entre sesgo y varianza en las estimaciones.

- Q(λ)

El algoritmo Q(λ) es una extensión del Q-learning que introduce el concepto de trazas de elegibilidad mediante el parámetro λ . Este mecanismo permite que el aprendizaje no solo dependa de la recompensa inmediata, sino también de múltiples pasos anteriores.

El parámetro λ regula la contribución de estos pasos anteriores:

- ▶ $\lambda = 0$ Se reduce a Q-learning tradicional (actualización basada solo en la recompensa inmediata).
- ▶ $\lambda = 1$ Se comporta como Monte Carlo (utiliza la recompensa total del episodio).
- ▶ $0 < \lambda < 1$ Realiza una interpolación entre ambos enfoques.

Puede requerir algún concepto de TD(λ)

- Regiones de Confianza

El objetivo de los métodos que se quieren analizar aquí es mejorar la estabilidad de la actualización de la política durante el entrenamiento. Existe un dilema: por un lado, nos gustaría entrenar lo más rápido posible, realizando grandes pasos durante la actualización mediante descenso de gradiente estocástico (SGD). Por otro lado, una actualización grande de la política generalmente es una mala idea. La política es algo muy no lineal, por lo que una actualización grande podría arruinar la política que acabamos de aprender.

La situación puede empeorar aún más en el ámbito del aprendizaje por refuerzo (RL), ya que no se puede recuperar de una mala actualización de la política mediante actualizaciones posteriores. En su lugar, la política incorrecta proporcionará muestras de experiencia deficientes que utilizaremos en pasos de entrenamiento posteriores, lo que podría romper nuestra política por completo. Por lo tanto, queremos evitar realizar actualizaciones grandes a toda costa. Una de las soluciones ingenuas sería utilizar una tasa de aprendizaje pequeña para dar pasos muy reducidos durante el SGD, pero esto ralentizaría significativamente la convergencia.

Para romper este círculo vicioso, varios investigadores han intentado estimar el efecto que tendrá nuestra actualización de la política en términos de resultados futuros. Constituyen los métodos de regiones de confianza (Trust Region Methods) en el aprendizaje por refuerzo. Son técnicas que se utilizan para garantizar que las actualizaciones de la política sean estables y no se produzcan cambios demasiado drásticos de una iteración a la siguiente. La idea central es limitar el tamaño del cambio de la política en cada actualización, de modo que la nueva política se mantenga lo suficientemente cercana a la anterior para no deteriorar el rendimiento aprendido.

TRPO (Trust Region Policy Optimization) es un algoritmo de actor-crítico. En los algoritmos de actor-crítico, la arquitectura consta de dos componentes principales:

- ▶ Actor: Se encarga de generar la política, es decir, determina las acciones a tomar dado un estado.
- ▶ Crítico: Evalúa la política generada por el actor calculando la función de valor (generalmente el valor de la acción o el valor del estado).

En TRPO, el actor es una red neuronal que aprende la política, mientras que el crítico estima la función de valor de la política. A lo largo del proceso de optimización, el actor y el crítico se actualizan conjuntamente.

Lo que distingue a TRPO de otros algoritmos actor-crítico es que utiliza una optimización basada en la *región de confianza*, lo que permite que los cambios en la política sean más estables y controlados, evitando actualizaciones demasiado grandes que puedan llevar a una degradación del rendimiento.

- Más Actores Críticos

Las técnicas de actor-crítico combinan dos componentes principales: el *actor*, que se encarga de seleccionar las acciones (política), y el *crítico*, que evalúa la calidad de las acciones o estados (función de valor). Existen numerosas variantes y mejoras en estos métodos. Cada uno de estos enfoques tiene sus propias ventajas y desafíos, y la elección del método adecuado puede depender de la naturaleza del problema, el tipo de espacio de acción (discreto o continuo) y las características específicas del entorno en el que se aplica. Cuando el espacio es continuo surgen nuevos retos.

DDPG (Deep Deterministic Policy Gradient) es un algoritmo de aprendizaje por refuerzo diseñado para resolver problemas con espacios de acción continuos, algo que otros algoritmos como DQN no pueden manejar directamente debido a que trabajan en espacios de acción discretos. DDPG se basa en el enfoque de gradientes de política determinista y utiliza dos redes neuronales una para el actor y otra para el crítico. Es una técnica off-policy.

Aunque DDPG puede lograr un gran rendimiento en algunas ocasiones, con frecuencia es sensible a los hiperparámetros y otros tipos de ajustes. Un modo común de falla en DDPG es que la función Q aprendida comienza a sobrestimar drásticamente los valores de Q, lo que lleva a que la política se rompa, ya que explota los errores en la función Q. Twin Delayed DDPG (TD3) es un algoritmo que aborda este problema mediante presenta un rendimiento sustancialmente mejorado en comparación con el DDPG base. Por algunos es considerado como casi insuperable (pero será cuestión de tiempo).

Puede desarrollar cualquiera de estos dos algoritmos (basta con uno).

- Otros trabajos por iniciativa propia

Alternativamente puede desarrollar un trabajo más aplicado. Por ejemplo, el uso del aprendizaje por refuerzo en el mundo de la robótica, videojuegos, etc ... Si le interesa trabajar sobre esta otra línea, diferente a las anteriores, remita al profesor la siguiente información: objetivo del trabajo, referencias documentales, librerías a utilizar y encaje y *dificultad* con el resto de los contenidos de la asignatura.

Otra alternativa, bastante menos investigadora, es la **construcción de una librería con todos los algoritmos** vistos en clase resolviendo algún entorno que pueda localizarse a partir de la web de Gymnasium.

La cantidad de trabajos sobre este tema es inabordable y requeriría una gran cantidad de tiempo para ponerse al día. OpenAI propone 105 artículos clave para este tema del aprendizaje por refuerzo publicados antes de 2019: <https://spinningup.openai.com/en/latest/spinningup/keypapers.html>. Si lo prefiere, también puede intentar reproducir algunos de los resultados que aparecen en esos artículos.

Hasta que no tenga el visto bueno para su realización, mejor no empiece. Su propuesta puede quedar rechazada.

! Importante

Recuerde que el objetivo es solo hacer un estudio de investigación. Solo debe estudiar uno o dos algoritmos de los que puede encontrar en la literatura y profundizar un poco en ellos. No se trata de que los estudie todos y decida por uno. Al contrario, decida uno y estúdielo (olvidando el resto de propuestas).

4. Referencias para la Investigación

A continuación se dan algunas citas bibliográficas de apoyo para la parte de investigación.

Bandidos Duelistas

1. Yue, Y., & Joachims, T. (2009). *Beat the mean bandit*. En **Proceedings of the 26th International Conference on Machine Learning (ICML)**. https://www.cs.cornell.edu/people/tj/publications/yue_joachims_11a.pdf Este trabajo introduce el problema del bandido duelista y propone algoritmos para abordarlo.
2. Zoghi, M., Whiteson, S., Munos, R., & de Rijke, M. (2014). *Relative upper confidence bound for the K-armed dueling bandit problem*. En **Proceedings of the 31st International Conference on Machine Learning (ICML)**. <https://proceedings.mlr.press/v32/zoghi14.html> Los autores proponen el algoritmo Relative Upper Confidence Bound (RUCB) para el problema del bandido duelista con K opciones.
3. Lekang, T., & Lamperski, A. (2019). *Simple Algorithms for Dueling Bandits*. En **arXiv preprint arXiv:1906.07611**. <https://arxiv.org/abs/1906.07611> Este artículo presenta algoritmos simples para el problema del bandido duelista, con análisis de sus límites de regret y comparaciones con algoritmos existentes.
4. González González, M. (2020). *Aprendizaje por refuerzo mediante bandidos duelistas*. Trabajo de Fin de Grado, Universidad de Sevilla. <https://miguelgg.com/assets/pdf/publications/TFGInfo.pdf> Este trabajo ofrece una visión detallada del problema de los bandidos duelistas y analiza diversos algoritmos, incluyendo aquellos de fácil implementación.

Bandidos Adversarios

1. Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002). *The nonstochastic multi-armed bandit problem*. **SIAM Journal on Computing**, 32(1), 48-77. Disponible en: <https://doi.org/10.1137/S0097539701398375> Dispone de una versión libre en <https://cseweb.ucsd.edu/~yfreund/papers/bandits.pdf> Este artículo introduce el algoritmo EXP3, diseñado para entornos adversariales en el problema del bandido multibrazo, y proporciona análisis teóricos sobre su rendimiento.
2. Lattimore, T., & Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge University Press. <https://doi.org/10.1017/9781108571401>. Disponible una versión libre en <https://tor-lattimore.com/downloads/book/book.pdf> Este libro estudia en la tercera parte el algoritmo EXP3.
3. Bubeck, S., & Cesa-Bianchi, N. (2012). *Regret analysis of stochastic and nonstochastic multi-armed bandit problems*. **Foundations and Trends in Machine Learning**, 5(1), 1-122. Disponible en: <https://doi.org/10.1561/22000000024> y también en <http://sbubeck.com/SurveyBCB12.pdf> Este trabajo ofrece una revisión exhaustiva de los análisis de regret en problemas de bandidos tanto estocásticos como adversariales, incluyendo algoritmos clave y sus propiedades.

4. Seldin, Y., & Slivkins, A. (2014). *One practical algorithm for both stochastic and adversarial bandits*. En **Proceedings of the 31st International Conference on Machine Learning (ICML)**. Disponible en: <https://proceedings.mlr.press/v32/seldin14.html> Los autores presentan un algoritmo práctico que es efectivo tanto en entornos estocásticos como adversales, adaptándose dinámicamente a la naturaleza del entorno.

Bandidos Contextuales

1. Lattimore, T., & Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge University Press. <https://doi.org/10.1017/9781108571401>. Disponible una versión libre en <https://tor-lattimore.com/downloads/book/book.pdf> Este libro estudia en la quinta parte el algoritmo EXP4
2. Bubeck, S., & Cesa-Bianchi, N. (2012). *Regret analysis of stochastic and nonstochastic multi-armed bandit problems*. **Foundations and Trends in Machine Learning**, 5(1), 1-122. Disponible en: <https://doi.org/10.1561/22000000024> y también en <http://sbubeck.com/SurveyBCB12.pdf> Este trabajo ofrece una revisión exhaustiva de los análisis de regret en problemas de bandidos tanto estocásticos como contextuales, incluyendo algoritmos clave y sus propiedades.
3. Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). *A Contextual-Bandit Approach to Personalized News Article Recommendation*. En **Proceedings of the 19th International Conference on World Wide Web (WWW)**. Disponible en: <https://doi.org/10.1145/1772690.1772758> y <https://arxiv.org/abs/1003.0146> Este trabajo presenta un enfoque de bandido contextual para la recomendación personalizada de artículos de noticias, introduciendo el algoritmo LinUCB y demostrando su eficacia en escenarios reales.
4. Zhou, L. (2015). *A Survey on Contextual Multi-armed Bandits*. **arXiv preprint arXiv:1508.03326**. Disponible en: <https://arxiv.org/abs/1508.03326> Este artículo presenta una revisión exhaustiva de los algoritmos de bandidos contextuales, abordando tanto enfoques estocásticos como adversariales, y proporcionando un análisis detallado sobre sus límites de arrepentimiento y suposiciones.
5. Chu, W., Li, L., Reyzin, L., & Schapire, R. E. (2011). *Contextual Bandits with Linear Payoffs*. En **Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)**. Disponible en: <https://proceedings.mlr.press/v15/chu11a.html> Este artículo introduce un enfoque basado en límites superiores de confianza (UCB) para bandidos contextuales con recompensas lineales, ofreciendo un equilibrio entre exploración y explotación y facilitando su implementación práctica.
6. Agrawal, S., & Goyal, N. (2013). *Thompson Sampling for Contextual Bandits with Linear Payoffs*. En **Proceedings of the 30th International Conference on Machine Learning (ICML)**. Disponible en: <https://proceedings.mlr.press/v28/agrawal13.html> Los autores analizan el uso de Thompson Sampling en el contexto de bandidos con recompensas lineales, proporcionando garantías teóricas sobre su rendimiento y comparándolo con otros algoritmos.

Planificación

1. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* Cap. 8 (2nd ed.). MIT Press. Disponible en: <http://incompleteideas.net/book/the-book-2nd.html>

TD(λ)

1. Sutton, R. S. (1988). *Learning to predict by the methods of temporal differences*. *Machine Learning*, 3(1), 9-44. Disponible en: <https://link.springer.com/article/10.1007/BF00115009> Este es el artículo seminal donde Richard Sutton introduce el método de Diferencia Temporal (TD) y presenta TD(λ) como una interpolación entre métodos TD y Monte Carlo.
2. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press. Disponible en: <http://incompleteideas.net/book/the-book-2nd.html> Este libro es una referencia fundamental sobre aprendizaje por refuerzo. Contiene explicaciones detalladas de TD(λ), trazas de elegibilidad y su aplicación en diferentes problemas de optimización.
3. Seijen, H. V., & Sutton, R. S. (2014). *True Online TD(λ)*. En **Proceedings of the 31st International**

- Conference on Machine Learning (ICML-14), 692-700. Disponible en: <http://proceedings.mlr.press/v32/seijen14.pdf> Este artículo introduce la versión *True Online TD(λ)*, una mejora sobre TD(λ) que hace que la actualización de valores sea más estable y eficiente en tiempo real.
4. van Seijen, H., & Sutton, R. S. (2014). *True Online TD(λ)*. En *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 692-700. Disponible en: <https://proceedings.mlr.press/v32/seijen14.pdf> Este artículo presenta el algoritmo True Online TD(λ), una variante que mejora la correspondencia entre la vista hacia adelante y la implementación en línea de TD(λ), ofreciendo actualizaciones más precisas y estables.
 5. van Seijen, H., Mahmood, A. R., Pilarski, P. M., Machado, M. C., & Sutton, R. S. (2016). *True Online Temporal-Difference Learning*. *Journal of Machine Learning Research*, 17(145), 1-40. Disponible en: <https://jmlr.org/papers/volume17/15-599/15-599.pdf> En este trabajo, los autores amplían el concepto de True Online TD(λ) a métodos de control, introduciendo True Online Sarsa(λ) y demostrando su eficacia en diversos dominios.

Q(λ)

1. Peng, J., & Williams, R. J. (1996). *Incremental Multi-Step Q-Learning*. En *Machine Learning*, 22(1-3), 283-290. Disponible en: <https://link.springer.com/article/10.1007/BF00114731> Este artículo introduce una versión de Q-learning con actualizaciones de múltiples pasos, precursora directa del algoritmo Q(λ).
2. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press. Disponible en: <http://incompleteideas.net/book/the-book-2nd.html> Este libro es una referencia fundamental en Aprendizaje por Refuerzo y proporciona una explicación detallada de Q(λ), sus propiedades y su relación con otros algoritmos.
3. Wiering, M. & Schmidhub, J. (s.f.). *Fast Online Q(λ)*. IDSIA, Corso Elvezia 36, 6. http://www.idsia.ch/~juergen/fast_online_q_lambda.pdf Este trabajo presenta una versión rápida y en línea del algoritmo Q(λ) para el aprendizaje por refuerzo, permitiendo actualizaciones eficientes en tiempo real. (La fecha de publicación no se encuentra especificada.)
4. Mousavi, S. S., Schukat, M., & Howley, E. (2017). *Applying Q(λ)-Learning in Deep Reinforcement Learning to Play Atari Games*. En *Proceedings of the Adaptive and Learning Agents Workshop (ALA 2017)*. Disponible en: https://ala2017.cs.universityofgalway.ie/papers/ALA2017_Mousavi.pdf Este trabajo explora la implementación de Q(λ) en un entorno de Aprendizaje Profundo aplicado a juegos de Atari, mostrando mejoras en la eficiencia del aprendizaje.
5. Sutton, R. S., Mahmood, A. R., Precup, D., & van Hasselt, H. (2014). *A new Q(λ) with interim forward view and Monte Carlo equivalence*. En *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 568-576. Disponible en: <https://proceedings.mlr.press/v32/sutton14.pdf> Este artículo presenta una nueva variante de Q(λ) que logra una equivalencia exacta con el enfoque de Monte Carlo, mejorando la precisión y estabilidad en el aprendizaje por refuerzo.
6. Harutyunyan, A., Bellemare, M. G., Stepleton, T., & Munos, R. (2016). *Q(λ) with Off-Policy Corrections*. Disponible en: <https://arxiv.org/abs/1602.04951> Los autores proponen y analizan un enfoque alternativo para el aprendizaje temporal-diferencial de múltiples pasos off-policy, introduciendo correcciones que mejoran la convergencia y eficiencia del algoritmo Q(λ).
7. Wiering, M. & Schmidhuber, J. (1998). *Fast Online Q(λ)*. *Machine Learning* Kluwer Academic Publishers, 33, 105-115. (1998) <https://link.springer.com/content/pdf/10.1023/A:1007562800292.pdf> Este artículo presenta una versión rápida y en línea del algoritmo Q(λ) para el aprendizaje por refuerzo, permitiendo actualizaciones eficientes en tiempo real.

Regiones de Confianza

1. <https://spinningup.openai.com/en/latest/algorithms/trpo.html>
2. Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). *Trust Region Policy Optimization*. En *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. <https://arxiv.org/abs/1502.05477>. Este artículo introduce TRPO, un algoritmo que mejora la

- estabilidad de las actualizaciones de la política al restringir la divergencia de Kullback-Leibler (KL) entre la política antigua y la nueva, garantizando pasos de actualización seguros.
3. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. <https://arxiv.org/abs/1707.06347> PPO simplifica la complejidad computacional de TRPO utilizando un mecanismo de *clipping* en la función objetivo para limitar la magnitud de las actualizaciones de la política, facilitando su implementación sin perder estabilidad.
 4. Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. <https://arxiv.org/abs/1506.02438> Este artículo introduce GAE (Generalized Advantage Estimation), que mejora la estimación de la ventaja reduciendo la varianza sin incurrir en un sesgo excesivo, facilitando actualizaciones de política más estables.
 5. Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). *Trust Region Policy Optimization*. <https://arxiv.org/abs/1502.05477> Aunque este artículo se centra en TRPO, incluye una discusión sobre GAE como parte integral del método para lograr actualizaciones de política confiables en entornos complejos.
 6. Wu, Y., Mansimov, E., Liao, S., Grosse, R., & Ba, J. (2017). *Scalable Trust-Region Method for Deep Reinforcement Learning Using Kronecker-Factored Approximation*. En **Proceedings of the 34th International Conference on Machine Learning (ICML)**. <https://arxiv.org/abs/1708.05144> ACKTR (Actor-Critic using Kronecker-Factored Trust Region) extiende la idea de las regiones de confianza a redes neuronales profundas, utilizando una aproximación factorizada de Kronecker para estimar la curvatura del espacio de parámetros (matriz de Fisher) y, de esta forma, realizar actualizaciones eficientes y estables en entornos de alta dimensión (redes neuronales profundas).
 7. Martens, J., & Grosse, R. (2015). *Optimizing Neural Networks with Kronecker-Factored Approximation*. <https://arxiv.org/abs/1503.05671> Aunque no es específico de RL, este trabajo proporciona la base teórica para la aproximación de la curvatura utilizada en ACKTR.
 8. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). *Asynchronous Methods for Deep Reinforcement Learning*. En **Proceedings of the 33rd International Conference on Machine Learning (ICML)**. <https://arxiv.org/abs/1602.01783> El A2C con línea base es una variante sincrónica del método Asynchronous Advantage Actor-Critic (A3C) que sirve como base estable y sencilla para métodos actor-crítico, facilitando la comparación con otros algoritmos porque ofrece un equilibrio adecuado entre exploración y explotación en entornos de acción continua.
 9. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning*. En **Proceedings of the 35th International Conference on Machine Learning (ICML)**. URL: <https://arxiv.org/abs/1801.01290> El método SAC (Soft Actor-Critic) no se clasifica (como A2C) como un método basado en regiones de confianza, aunque SAC comparte con estos métodos el objetivo de lograr estabilidad durante la actualización de la política, lo hace mediante mecanismos diferentes. SAC se basa en un objetivo de entropía máxima que incentiva la exploración y la robustez en la toma de decisiones. La estabilidad en SAC se logra a través de la incorporación de un término de entropía en la función de objetivo, lo que suaviza la política y fomenta actualizaciones más estables. No utiliza restricciones explícitas en la forma de una región de confianza como PPO o TRPO; pero como "comparten el mismo objetivo" son candidatos a ser comparables.

Más Actores Críticos

1. Algoritmo DDPG: <https://spinningup.openai.com/en/latest/algorithms/ddpg.html>
2. Algoritmo TD3: <https://spinningup.openai.com/en/latest/algorithms/td3.html>
3. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ..., & Kavukcuoglu, K. (2016). *Asynchronous Methods for Deep Reinforcement Learning*. <https://arxiv.org/abs/1602.01783> Este artículo introduce A3C, cuyo modo sincrónico se conoce como A2C, utilizando múltiples agentes para actualizar la política y reducir la varianza en las actualizaciones.
4. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ..., & Hassabis, D.

- (2015). *Human-level Control through Deep Reinforcement Learning*. <https://www.nature.com/articles/nature14236> Aunque se centra en DQN, este trabajo sienta las bases para los métodos actor-crítico al demostrar la eficacia de las aproximaciones basadas en redes neuronales profundas en RL.
5. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). *Continuous Control with Deep Reinforcement Learning*. <https://arxiv.org/abs/1509.02971> Este artículo introduce DDPG, un método off-policy para control continuo que combina un actor determinista con un crítico basado en Q.
 6. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2017). *Deep Reinforcement Learning That Matters*. <https://arxiv.org/abs/1709.06560> Este trabajo analiza comparativamente varios algoritmos de RL, incluyendo DDPG, y destaca buenas prácticas y desafíos en su implementación.
 7. Fujimoto, S., van Hoof, H., & Meger, D. (2018). *Addressing Function Approximation Error in Actor-Critic Methods*. <https://arxiv.org/abs/1802.09477> Este artículo introduce TD3, que mejora DDPG mitigando el error de aproximación de funciones mediante el uso de dos críticos y actualizaciones retrasadas del actor.
 8. Fujimoto, S., van Hoof, H., & Meger, D. (2018). *TD3: Twin Delayed Deep Deterministic Policy Gradient*. <https://openreview.net/forum?id=rJk00ekbF> Versión publicada de TD3, que detalla las modificaciones implementadas para mejorar la estabilidad y el rendimiento en tareas de control continuo.
 9. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning*. <https://arxiv.org/abs/1801.01290> Este artículo introduce SAC, un método off-policy que utiliza un objetivo de entropía máxima para fomentar la exploración y mejorar la estabilidad en el entrenamiento.
 10. Haarnoja, T., Zhou, A., Tang, H., & Levine, S. (2018). *Soft Actor-Critic Algorithms and Applications*. <https://arxiv.org/abs/1812.05905> Este trabajo extiende SAC y explora diversas aplicaciones, proporcionando una visión más amplia de su desempeño en entornos de control continuo.
 11. Wang, Z., Schaul, T., Hessel, M., Hasselt, H. V., Lanctot, M., & Freitas, N. (2017). *Sample Efficient Actor-Critic with Experience Replay*. <https://arxiv.org/abs/1611.01224> Este artículo introduce ACER, que integra el replay de experiencia en el marco actor-crítico para mejorar la eficiencia muestral y la estabilidad del entrenamiento.
 12. Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., & Dabney, W. (2018). *Recurrent Experience Replay in Distributed Reinforcement Learning*. <https://arxiv.org/abs/1803.07240> Este trabajo propone mejoras al replay de experiencia en entornos distribuidos, complementando y extendiendo ideas presentes en ACER.

5. Uso básico de Gymnasium

Gymnasium (<https://gymnasium.farama.org/>) es un entorno que permite ensayar con distintos algoritmos de aprendizaje por refuerzo. Se muestra a continuación algunos de trozos de código básicos.

Generar un episodio

```
import gymnasium as gym
env = gym.make('nombreEntorno')

obs, info = env.reset() # Genera un estado inicial

done = False
while not done: # Genera muestras hasta generar el episodio
    action = env.action_space.sample() # Selecciona una acción aleatoria
    next_obs, reward, terminated, truncated, info = env.step(action) # Ejecuta la acción
    done = terminated or truncated # Determina si se terminó.
    obs = next_obs # Pasa al siguiente estado
```

Esquema general de un agente

```
class Agent:
    def __init__(self, env:gym.Env, hiperparámetros):
        """Inicializa todo lo necesario para el aprendizaje"""

    def get_action(self, state) -> Any:
        """
        Indicará qué acción realizar de acuerdo al estado.
        Responde a la política del agente.
        Construir tantas funciones de este tipo como políticas se quieran usar.
        """

    def update(self, obs, action, next_obs, reward, terminated, truncated, info):
        """
        Con la muestra (s, a, s', r) e información complementaria aplicamos el algoritmo.
        update() no es más que el algoritmo de aprendizaje del agente.
        Se añadirá lo necesario para obtener resultados estadísticos, evolución, etc ...
        """
```

Esquema general del aprendizaje episódico de un agente

```
import gymnasium as gym
from agents import AgentMonteCarlo

env = gym.make('nombreEntorno')

agent = AgentMonteCarloOnPolicy(env) # Si queremos usa MC

for episode in tqdm(range(n_episodes)):
    obs, info = env.reset()
    done = False

    # play one episode
    while not done:
        action = agent.get_action(obs)
        next_obs, reward, terminated, truncated, info = env.step(action)

        # update the agent
        agent.update(self, obs, action, next_obs, reward, terminated, truncated, info)

        # update if the environment is done and the current obs
        done = terminated or truncated
        obs = next_obs

stats = agent.stats() # Retorna los resultados estadísticos, evolución, etc ...

# Mostrar las estadísticas
```

Espacios

Todo entorno tiene un espacio de acciones (acciones válidas) y un espacio de estados (observaciones válidas).

```
print(env.action_space, env.observation_space)
```

Todos los espacios heredan de la superclase Space. Hay varios tipos de espacios (<https://gymnasium.farama.org/api/spaces/>). Destacamos:

- Discrete: representa un conjunto de elementos mutuamente excluyentes, numerados de 0 a $n - 1$. `env.action_space.n` retorna un entero con el número de elementos del espacio. Por ejemplo, `Discrete(n=4)` puede usarse para un espacio de acción de cuatro direcciones para moverse [izquierda, derecha, arriba o abajo].

- **Box**: representa un tensor de n dimensiones de números racionales con intervalos $[low, high]$. Por ejemplo, un pedal acelerador con un solo valor entre 0.0 y 1.0 podría codificarse como `Box(low=0.0, high=1.0, shape=(1,), dtype=np.float32)` (el argumento `shape` es una tupla de longitud 1 con un solo valor de 1, lo que nos da un tensor unidimensional con un solo valor). Otro ejemplo de **Box** podría ser una observación de pantalla de Atari que es una imagen RGB de tamaño 210×160 : `Box(low=0, high=255, shape=(210, 160, 3), dtype=np.uint8)`. En este caso, el argumento `shape` es una tupla de tres elementos: la primera dimensión es la altura de la imagen, la segunda es el ancho y la tercera es igual a 3, que corresponden a los tres planos de color para rojo, verde y azul, respectivamente. Entonces, en total, cada observación es un tensor 3D con 100,800 bytes.
- **Tuple**: nos permite combinar varias instancias de la clase `Space`. Esto nos permite crear espacios de acción y observación de cualquier complejidad que deseemos. Por ejemplo, imagina que queremos crear una especificación de espacio de acción para un automóvil. El automóvil tiene varios controles que podrían cambiarse en cada instante, incluyendo el ángulo del volante, la posición del pedal del freno y la posición del pedal del acelerador. Estos tres controles podrían especificarse mediante tres valores flotantes en una sola instancia de **Box**. Además de estos controles esenciales, el automóvil tiene controles discretos adicionales, como una señal de giro (que podría ser "apagada", "derecha" o "izquierda"), bocina ("encendida" o "apagada"), entre otros. Para combinar todo esto en una sola clase de especificación de espacio de acción, podemos crear `Tuple(spaces=(Box(low=-1.0, high=1.0, shape=(3,), dtype=np.float32), Discrete(n=3), Discrete(n=2)))`.

Hay más tipos de espacios. Consulte <https://gymnasium.farama.org/api/spaces/#fundamental-spaces>

Wrapper

En ocasiones se quiere extender la funcionalidad del entorno de alguna manera genérica. Por ejemplo, un entorno te proporciona algunas observaciones, pero deseas acumularlas en algún buffer y proporcionar al agente las últimas N observaciones.

Hay muchas situaciones de este tipo: te gustaría "envolver" el entorno existente y añadir alguna lógica extra que haga algo más. Gymnasium proporciona un marco general para estas situaciones con la clase `Wrapper`.

La clase `Wrapper` hereda de la clase `Env`. Su constructor acepta un solo argumento: la instancia de la clase `Env` que se va a "envolver". Para añadir funcionalidad extra, necesitas redefinir los métodos que deseas extender, como `step()` o `reset()`. El único requisito es llamar al método original de la superclase.

Gymnasium proporciona muchas subclases de la clase `Wrapper` (<https://gymnasium.farama.org/api/wrappers/table/>). Comentamos algunas que permiten filtrar solo una parte específica de la información.

- **ObservationWrapper**: Necesitas redefinir el método `observation(obs)` de la clase padre. El argumento `obs` es una observación del entorno envuelto, y este método debe devolver la observación que se le dará al agente.
- **RewardWrapper**: Esta supone modificar el método `reward(rew)`, que podría modificar el valor de la recompensa dada al agente.
- **ActionWrapper**: Necesitas sobrecribir el método `action(act)`, que podría ajustar la acción pasada al entorno envuelto para el agente.

Para hacerlo un poco más práctico, imaginemos una situación en la que quiere aplicar una política ϵ -greedy, entonces una forma de hacerlo es:

```
class RandomActionWrapper(ActionWrapper):
    def __init__(self, env, epsilon=0.1):
        super().__init__(env)
```

```
self.epsilon = epsilon

def action(self, action):
    if random.random() < self.epsilon:
        return self.env.action_space.sample()
    return action
```

Notar que en este ejemplo se delega la política del agente en el entorno. Es decir, estamos modificando el comportamiento del entorno con este Wrapper. Se aconseja que las políticas se incluyan mejor en el agente.

Hay un Wrapper que permite grabar vídeos de los episodios que se generen en el entorno usando el método `render()`. No se recomienda su uso **salvo que quiera mostrar** de forma animada **la política óptima final obtenida después del entrenamiento**. En https://github.com/ldaniel-hm/eml_tabular/blob/main/EjemploGeneracionVideos.ipynb tiene un ejemplo de como usarlo desde Colab.