# Marco teórico:
# Aprendizaje por Refuerzo

*Jaime Espel Manzano*
2024-10-16

# Resumen

Los capítulos que se presentan a continuación, desde el Capítulo 1 hasta el 7 constituyen un resumen basado en el libro *Reinforcement Learning: An Introduction* (Sutton, Barto, 2018) [1]. Cabe destacar que todas las figuras y algoritmos presentados en este capítulo provienen de dicha obra. A lo largo de estos capítulos se abordan los principales conceptos descritos en el libro, con el fin de proporcionar una base sólida del Reinforcement Learning. Finalmente, en el Capítulo 8 se ofrece la explicación del *Transfer Learning*.

# Índice de contenidos

# Índice de figuras

# Lista de Algoritmos

CAPÍTULO 1

# Introducción

El aprendizaje por refuerzo es aprender qué hacer —cómo mapear situaciones a acciones— con el fin de maximizar una señal de recompensa numérica. A diferencia de los métodos de aprendizaje tradicionales, al agente no se le dice explícitamente qué acciones tomar; en su lugar, debe descubrir qué acciones producen la mayor recompensa probándolas. En los casos más interesantes y desafiantes, las acciones no solo afectan las recompensas inmediatas, sino que también influyen en los estados futuros y, a través de ellos, todas las recompensas posteriores.

## 1.1. Contexto en el Machine Learning

El aprendizaje por refuerzo es uno de los tres principales paradigmas del aprendizaje automático (Machine Learning), junto con el aprendizaje supervisado y el no supervisado.

El *aprendizaje supervisado* consiste en aprender a partir de un conjunto de ejemplos etiquetados proporcionados por un supervisor externo. Cada ejemplo es una descripción de una situación junto con una especificación, la etiqueta, que indica la categoría (en la regresión logística) o el valor (en la regresión lineal) correspondiente a la situación. El objetivo de este tipo de aprendizaje es que el sistema extrapole o generalice sus respuestas para que actúe correctamente en situaciones que no están presentes en el conjunto de entrenamiento.

En problemas interactivos, a menudo es poco práctico obtener ejemplos de comportamiento deseado que sean correctos y representativos de todas las situaciones en las que el agente tiene que actuar. Por ello, el aprendizaje por refuerzo se presenta como una solución ventajosa ante estos.

El *aprendizaje no supervisado* se centra principalmente en descubrir estructuras ocultas en colecciones de datos no etiquetados. Aunque descubrir tales estructuras puede ser valioso en el aprendizaje por refuerzo, por sí solo no resuelve el reto principal de este tipo de aprendizaje, que es maximizar una señal de recompensa.

Uno de los desafíos que surgen en el aprendizaje por refuerzo, y no en otros tipos de aprendizaje, es el equilibrio entre exploración y explotación.

Para obtener buenas recompensas, un agente de aprendizaje por refuerzo debe preferir las acciones que ha probado en el pasado y que ha encontrado eficaces en la obtención de recompensas. Pero, para descubrir tales acciones, tiene que intentar acciones que no ha seleccionado antes. El agente tiene que *explotar* lo que ya ha experimentado para obtener mayores recompensas, pero también tiene que *explorar* para hacer mejores selecciones de acciones en el futuro.

El dilema radica en que ni la exploración ni la explotación pueden ser perseguidas exclusivamente sin fracasar en la tarea. El agente debe probar distintas acciones y favorecer progresivamente aquellas que parecen ser las mejores.

## 1.2. Elementos del Aprendizaje por Refuerzo

Un sistema de aprendizaje por refuerzo consta de dos componentes fundamentales:

- **Agente:** El *agente* es la entidad que toma decisiones y aprende en función de las interacciones con su entorno.
- **Entorno:** El *entorno* es todo aquello que rodea al agente, incluyendo las condiciones y reglas bajo las cuales el agente opera.

Más allá del agente y del entorno, se pueden identificar cuatro subelementos principales en un sistema de aprendizaje por refuerzo:

- **Política:** Una *política* define la forma en que el agente de aprendizaje se comporta en un momento dado. En términos generales, una política es un mapeo de los estados percibidos del entorno a las acciones que se deben tomar en esos estados. Corresponde a lo que en psicología se llamaría un conjunto de reglas o asociaciones de estímulo-respuesta.
- **Señal de recompensa:** Una *señal de recompensa* define el objetivo de un problema de aprendizaje por refuerzo. En cada paso de tiempo, el entorno envía al agente de aprendizaje por refuerzo un número único llamado recompensa. La señal de recompensa define así cuáles son los eventos buenos y malos para el agente.
- **Función de valor:** En términos generales, el *valor* de un estado es la cantidad total de recompensa que un agente puede esperar acumular en el futuro, comenzando desde ese estado, es decir, una *función de valor* especifica lo que es bueno a largo plazo.

Mientras que las recompensas determinan la deseabilidad intrínseca e inmediata de los estados del entorno, los valores indican la deseabilidad *a largo plazo* de los estados, teniendo en cuenta los estados que probablemente seguirán y las recompensas disponibles en esos estados. Por ejemplo, un estado podría siempre producir una baja recompensa inmediata, pero aún así tener un alto valor porque regularmente es seguido por otros estados que producen altas recompensas. O lo contrario.

Para hacer una analogía humana, las recompensas son algo así como el placer (si son altas) y el dolor (si son bajas), mientras que los valores corresponden a un juicio más refinado y previsor de cuán complacidos o descontentos estamos de que nuestro entorno esté en un estado particular.

Desafortunadamente, es mucho más difícil determinar los valores que determinar las recompensas. Básicamente, las recompensas vienen dadas directamente por el entorno, pero los valores deben estimarse y reestimarse a partir de las secuencias de observaciones que realiza un agente a lo largo de toda su vida. De hecho, el componente más importante de casi todos los algoritmos de aprendizaje por refuerzo es un método para estimar los valores de manera eficiente.

- **Modelo:** Un *modelo* es algo que imita el comportamiento del entorno o, de manera más general, que permite hacer inferencias sobre cómo se comportará el entorno.

Los modelos se utilizan para la *planificación*, por lo que nos referimos a cualquier forma de decidir un curso de acción al considerar posibles situaciones futuras antes de que realmente se experimenten. Los métodos para resolver problemas de aprendizaje por refuerzo que utilizan modelos y planificación se llaman métodos *basados en modelos*, en oposición a los métodos más simples *sin modelos* que son explícitamente aprendices de prueba y error.

CAPÍTULO 2

# Multi-armed Bandits

La característica más importante que distingue el aprendizaje por refuerzo de otros tipos de aprendizaje es que evalúa las acciones tomadas en lugar de instruir dando las acciones correctas. El feedback puramente evaluativo indica lo buena que ha sido la acción tomada, pero no si ha sido la mejor o la peor acción posible. Esto crea la necesidad de una exploración activa en búsqueda del buen comportamiento.

En este capítulo estudiaremos el aspecto evaluativo del aprendizaje por refuerzo en un escenario simplificado, que proporcionará una buena base para comprender diversos aspectos del RL. El problema concreto que estudiamos es una versión simplificada del problema *k-armed bandit*, en el que solo se toman decisiones en una única situación, ya que las acciones seleccionadas no influyen en el entorno del agente.

## 2.1. El problema *k*-armed Bandit

En este problema el agente se enfrenta repetidamente a una elección entre $k$ acciones diferentes. Después de cada elección, recibe una recompensa numérica elegida a partir de una distribución de probabilidad estacionaria que depende de la acción que seleccione. El objetivo es maximizar la recompensa total esperada durante un período de tiempo, por ejemplo, a lo largo de 1000 selecciones de acciones.

El nombre *k-armed Bandit* proviene de las máquinas tragaperras (conocidas en inglés como "one-armed bandits" debido a su palanca única), salvo en este caso que tiene $k$ palancas en lugar de una.

En nuestro problema $k$-armed bandit, cada una de las $k$ acciones tiene una recompensa esperada, o recompensa promedio, cuando se selecciona esa acción; llamémosle el *valor* de esa acción. Denotamos la acción seleccionada en el paso de tiempo $t$ como $A_t$, y la recompensa correspondiente como $R_t$. El valor de una acción arbitraria $a$, denotado como $q_*(a)$, es la recompensa esperada dado que se selecciona $a$:

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a] \,.$$

Si el agente conociera el valor de cada acción, sería trivial resolver el problema: el agente seleccionaría siempre la acción con mayor valor. Suponemos que el agente no conoce con certeza los valores de las acciones, luego tiene que estimarlos. Denotamos el valor estimado de la acción $a$ en el paso de tiempo $t$ como $Q_t(a)$. Nos gustaría que $Q_t(a)$ fuese lo más parecido posible a $q_*(a)$.

## 2.2. Métodos de Valor de Acción

Los métodos de valor de acción implican estimar los valores de las acciones y utilizar estas estimaciones para tomar decisiones sobre qué acciones realizar.

En este problema, el valor verdadero de una acción es la recompensa media cuando se selecciona esa acción. Una forma natural de estimar esto es promediando las recompensas recibidas:

$$Q_t(a) \doteq \frac{\text{suma de recompensas cuando se eligió } a \text{ antes de } t}{\text{número de veces que se eligió } a \text{ antes de } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1} A_i = a}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i = a}}, \quad (2.1)$$

donde $\mathbb{1}_{predicado}$ denota la variable aleatoria que toma el valor 1 si el predicado es verdadero y 0 si no lo es.

Si el denominador es cero, entonces definimos $Q_t(a)$ con un valor por defecto, como 0. A medida que el denominador tiende a infinito, por la ley de los grandes números, $Q_t(a)$ converge a $q_*(a)$. Esto se denomina el método *sample-average* para estimar los valores de las acciones, ya que cada estimación es un promedio de la muestra de recompensas. Por supuesto, este es solo uno de los métodos para estimar los valores de las acciones, y no necesariamente el mejor.

La regla más simple para la selección de acciones es elegir una de las acciones con el valor estimado más alto, es decir, una de las acciones codiciosas (en inglés, *greedy*). Si hay más de una acción codiciosa, entonces se realiza una selección entre ellas de alguna manera arbitraria, en su defecto al azar. Escribimos este método de selección como

$$A_t \doteq \arg\max_a Q_t(a), \quad (2.2)$$

donde $\arg\max_a$ denota la acción $a$ para la que se maximiza la expresión siguiente.

La selección codiciosa de acciones siempre explota el conocimiento actual para maximizar la recompensa inmediata. Sin embargo, si un agente es demasiado codicioso y sólo explota, podría no aprender estimaciones precisas de valor. Si un agente tiene estimaciones de valor imprecisas, puede seleccionar una acción subóptima cuando intenta explotar. Por eso, la exploración, que implica sacrificar la recompensa inmediata al seleccionar acciones no codiciosas para obtener mejores estimaciones de valor, es crucial a la hora de seleccionar acciones.

Una alternativa sencilla es comportarse de forma codiciosa la mayor parte del tiempo, pero de vez en cuando, digamos con una pequeña probabilidad $\varepsilon$, seleccionar al azar entre todas las acciones con igual probabilidad, independientemente de las estimaciones del valor de las acciones. Llamamos a los métodos que utilizan esta regla de selección métodos *$\varepsilon$-greedy*. Entonces, la probabilidad de seleccionar la acción codiciosa es $1 - \varepsilon + \frac{\varepsilon}{k}$ y para las acciones no codiciosas es $\frac{\varepsilon}{k}$.

Una ventaja de este tipo de métodos es que, en el límite a medida que aumenta el número de pasos, cada acción será muestreada un número infinito de veces, asegurando así que todos los $Q_t(a)$ convergen a $q_*(a)$.

## 2.3. Implementación Incremental

Veamos cómo se calculan los promedios, utilizados en los métodos de valor de acción, de manera eficiente. Centrándonos en una sola acción, sea $R_i$ la recompensa recibida tras la $i$ésima selección de esta acción, y sea $Q_n$ la estimación de su valor de acción tras haber sido seleccionada $n-1$ veces, que podemos escribir simplemente como

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

La implementación más directa sería mantener un registro de todas las recompensas y luego realizar este cálculo cada vez que se necesite el valor estimado. Sin embargo, si se hace esto, los requisitos de memoria y computación crecerían con el tiempo a medida que se vieran más recompensas. Como se puede sospechar, esto no es realmente necesario. Puede calcularse de la siguiente manera:

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]. \qquad (2.3)
\end{aligned}$$

Esta implementación requiere solo la memoria para $Q_n$ y $n$, y solo el pequeño cálculo en (2.3) para cada nueva recompensa.

Esta regla de actualización (2.3) tiene una forma que aparece con frecuencia en los algoritmos de RL. La forma general es

$$\textit{NuevaEstimación} \leftarrow \textit{EstimaciónPrevia} + \textit{TamañoDePaso}\left[\textit{Objetivo} - \textit{EstimaciónPrevia}\right]. \quad (2.4)$$

La expresión $\left[\textit{Objetivo} - \textit{EstimaciónPrevia}\right]$ es un error en la estimación, que se reduce dando un paso hacia el "Objetivo". El tamaño de paso (*StepSize*) en el método incremental (2.3) varía con cada paso de tiempo, utilizando $\frac{1}{n}$ al procesar la recompensa $n$ésima para la acción $a$. Este parámetro se denota como $\alpha$ o, más generalmente, por $\alpha_t(a)$.

## 2.4. Valores Iniciales Optimistas

Los métodos de valor de acción dependen en cierta medida de las estimaciones iniciales del valor de la acción, $Q_1(a)$. En el lenguaje estadístico, estos métodos están sesgados por sus estimaciones iniciales. En la práctica, este tipo de sesgo no suele ser un problema y a veces puede ser muy útil. El inconveniente es que las estimaciones iniciales se convierten, en un conjunto de parámetros que el usuario debe elegir, aunque sólo sea para ponerlos todos a cero. La ventaja es que nos permite proporcionar algún conocimiento previo sobre el problema.

Los valores de acción iniciales también se pueden usar como una forma simple de fomentar la exploración. Supongamos que en lugar de establecer los valores de acción iniciales en cero, los establecemos todos en +5. Si los $q_*(a)$ en este problema se seleccionan de una distribución normal con media 0 y varianza 1, una estimación inicial de +5 es extremadamente optimista. El resultado es que se prueban todas las acciones varias veces antes de que las estimaciones de valor converjan. El sistema realiza una cantidad considerable de exploración, incluso si se seleccionan acciones codiciosas todo el tiempo.

Esta técnica para fomentar la exploración se llama *valores iniciales optimistas*. Es un truco simple que puede ser bastante efectivo en problemas estacionarios, donde las condiciones y las recompensas no cambian con el tiempo, pero no es adecuado para problemas no estacionarios, donde se necesitaría un impulso continuo para explorar nuevas opciones.

CAPÍTULO 3

# Procesos de Decisión de Markov Finitos

Los procesos de decisión de Markov finitos, o MDP finitos (por sus siglas en inglés, *Markov Decision Processes*), son una formalización clásica de la toma de decisiones secuencial. En los MDP finitos, las acciones influyen no solo en las recompensas inmediatas, sino también en las situaciones subsiguientes, o estados, y a través de estos en las recompensas futuras. El objetivo del agente es maximizar la recompensa acumulada esperada en el futuro.

## 3.1. La Interfaz Agente-Entorno

Los MDP pretenden ser un marco directo para el problema de aprender a partir de la interacción para lograr un objetivo. El aprendiz y tomador de decisiones se llama *agente*. Lo que interactúa con él, que abarca todo lo que está fuera del agente, se llama el *entorno*. Estos interactúan continuamente, el agente selecciona acciones y el entorno responde a estas acciones y le presenta nuevas situaciones al agente. El entorno también genera recompensas, valores numéricos especiales que el agente trata de maximizar a lo largo del tiempo a través de su elección de acciones. La Figura 3.1 de *Reinforcement Learning: An Introduction* (Sutton, Barto, 2018) ilustra la interacción entre el agente y el entorno.

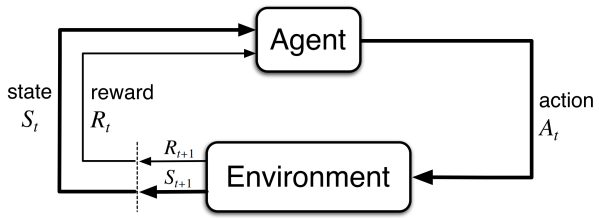



**Figura 3.1:** La interacción agente-entorno en un proceso de decisión de Markov

En cada paso de tiempo $t = 0, 1, 2, 3, \ldots$, el agente recibe alguna representación del estado del entorno, $S_t \in \mathcal{S}$, y en base a ello selecciona una acción, $A_t \in \mathcal{A}(s)$. Un paso de tiempo después, en parte como consecuencia de su acción, el agente recibe una recompensa numérica, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, y se encuentra en un nuevo estado, $S_{t+1}$. El MDP y el agente juntos dan lugar a una secuencia o trayectoria que comienza así:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \tag{3.1}$$

En un MDP *finito*, los conjuntos de estados, acciones y recompensas ($\mathcal{S}$, $\mathcal{A}$, y $\mathcal{R}$) tienen un número finito de elementos.

En un proceso de decisión de Markov, las variables aleatorias $R_t$ y $S_t$ tienen distribuciones de probabilidad discretas bien definidas que dependen únicamente del estado y la acción precedentes. En concreto, para cualquier paso de tiempo $t$, la probabilidad de transitar a un estado posterior $S_t$ y recibir una recompensa $R_t$ depende únicamente del estado anterior $S_{t-1}$ y de la acción $A_{t-1}$ tomada por el agente. Formalmente, esto se expresa como:

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}, \tag{3.2}$$

para todos $s' \in \mathcal{S}$, $s \in \mathcal{S}$, $r \in \mathcal{R}$, y $a \in \mathcal{A}(s)$.

La función de dinámica $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ es una función determinista ordinaria de cuatro argumentos. A partir de esta función, se puede calcular cualquier otra cosa que uno pueda querer saber sobre el entorno, como:

- Las *probabilidades de transición de estado* (que denotamos, con un ligero abuso de notación, como una función de tres argumentos $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$):

$$p(s' \mid s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a). \tag{3.3}$$

- Las *recompensas esperadas para pares estado–acción* como una función de dos argumentos $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a), \tag{3.4}$$

- Las *recompensas esperadas para triples estado–acción–siguiente estado* como una función de tres argumentos $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$:

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}. \tag{3.5}$$

## 3.2. Ejemplo: Robot de Reciclaje

Un robot móvil tiene la tarea de recoger latas vacías de refresco en un entorno de oficina. Tiene sensores para detectar latas, y un brazo con una pinza que puede recogerlas y colocarlas en un contenedor a bordo; funciona con una batería recargable.

Suponemos dos niveles de carga, alto y bajo: $\mathcal{S} = \{\texttt{high}, \texttt{low}\}$. En cada estado, el agente puede decidir si (1) buscar activamente una lata durante un cierto período de tiempo, `buscar` o `search`, (2) permanecer estacionario y esperar a que alguien le traiga una lata, `esperar` o `wait`, o (3) regresar a su base para recargar la batería, `recargar` o `recharge`. Cuando el nivel de energía es alto, recargar no es necesario, por lo que las acciones son $\mathcal{A}(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$ y $\mathcal{A}(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$.

Las recompensas son cero la mayor parte del tiempo, pero se vuelven positivas cuando el robot asegura una lata vacía, o negativas si la batería se agota por completo.

La mejor forma de encontrar latas es buscarlas activamente, pero esto agota la batería del robot, mientras que esperar no lo hace. Buscar activamente una lata con energía alta deja la batería alta con probabilidad $\alpha$ y la reduce a baja con probabilidad $1 - \alpha$. Con energía baja, buscar activamente una lata mantiene la batería baja con probabilidad $\beta$ y la agota con probabilidad $1 - \beta$. Rescatar al robot tras agotar la batería da una recompensa de $-3$. Las recompensas esperadas para buscar y esperar son $r_{\texttt{search}}$ y $r_{\texttt{wait}}$, con $r_{\texttt{search}} > r_{\texttt{wait}}$, que denotan el número esperado de latas que el robot recolectará.

Este sistema es entonces un MDP finito, y podemos escribir las probabilidades de transición y las recompensas esperadas, con la dinámica indicada en la Figura 3.2:

| $s$ | $a$ | $s'$ | $p(s' \mid s, a)$ | $r(s, a, s')$ |
|---|---|---|---|---|
| high | search | high | $\alpha$ | $r_{\texttt{search}}$ |
| high | search | low | $1 - \alpha$ | $r_{\texttt{search}}$ |
| low | search | high | $1 - \beta$ | $-3$ |
| low | search | low | $\beta$ | $r_{\texttt{search}}$ |
| high | wait | high | 1 | $r_{\texttt{wait}}$ |
| high | wait | low | 0 | - |
| low | wait | high | 0 | - |
| low | wait | low | 1 | $r_{\texttt{wait}}$ |
| low | recharge | high | 1 | 0 |
| low | recharge | low | 0 | - |

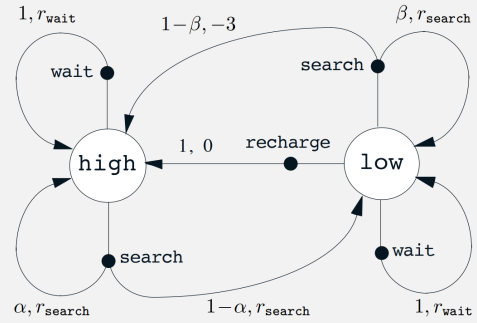



**Figura 3.2:** Robot de Reciclaje: Probabilidades de transición y recompensas

A la derecha se muestra otra forma útil de resumir la dinámica de un MDP finito, como un *grafo de transición*. Hay dos tipos de nodos: *nodos de estado* y *nodos de acción*. Hay un nodo de estado para cada posible estado (un gran círculo abierto etiquetado con el nombre del estado) y un nodo de acción para cada par estado-acción (un pequeño círculo sólido etiquetado con el nombre de la acción y conectado por una línea al nodo de estado).

Cabe destacar que las probabilidades de transición que etiquetan las flechas que salen de un nodo de acción siempre suman 1.

## 3.3. Objetivos y Recompensas

En el aprendizaje por refuerzo, el propósito u objetivo del agente se formaliza en términos de una señal especial, llamada *recompensa*, que pasa del entorno al agente. En cada paso de tiempo, la recompensa es un número simple, $R_t \in \mathbb{R}$.

El agente siempre aprende a maximizar su recompensa. Por lo tanto, es fundamental que las recompensas que establezcamos indiquen realmente lo que queremos lograr. En particular, la señal de recompensa no es el lugar para impartir al agente conocimiento previo sobre *cómo* lograr lo que queremos que haga. La señal de recompensa es la forma de comunicar al agente lo que se desea conseguir.

Por ejemplo, un agente que juega al ajedrez debería ser recompensado solo por ganar, no por lograr subobjetivos como tomar las piezas de su oponente. Si se recompensara el logro de este tipo de subobjetivos, entonces el agente podría encontrar una manera de lograrlos sin alcanzar el objetivo real. Por ejemplo, podría encontrar una forma de tomar las piezas del oponente incluso a costa de perder la partida.

Para que un agente aprenda a jugar ajedrez, las recompensas naturales son +1 por ganar, -1 por perder, y 0 por empatar y para todas las posiciones no terminales.

## 3.4. Retornos y Episodios

El objetivo del aprendizaje es maximizar la recompensa acumulada que se recibe a largo plazo. Si la secuencia de recompensas recibidas tras el paso de tiempo $t$ se denota $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$, entonces buscamos maximizar el *retorno esperado*, donde el retorno, denotado $G_t$, se define como alguna función específica de la secuencia de recompensas.

En el caso más sencillo, el retorno es la suma de las recompensas:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T, \tag{3.6}$$

donde $T$ es un paso de tiempo final. Este enfoque tiene sentido en aplicaciones en las que existe una noción natural del paso de tiempo final, las tareas de este tipo se denominan *tareas episódicas*.

Por otra parte, en muchos casos la interacción agente-entorno continúa sin límite. Las denominamos *tareas continuas*. La fórmula de retorno (3.6) es problemática para las tareas continuas porque el paso de tiempo final sería $T = \infty$, y el retorno, que es lo que estamos tratando de maximizar, podría divergir fácilmente.

Para hacer converger $G_t$ añadimos un parámetro $\gamma$ , $0 \leq \gamma \leq 1$, llamado *tasa de descuento*. Según este enfoque, el agente intenta seleccionar acciones de modo que se maximice la suma de las recompensas descontadas que recibe en el futuro. En concreto, elige $A_t$ para maximizar el *rendimiento descontado* esperado:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{3.7}$$

La tasa de descuento determina el valor actual de las recompensas futuras: una recompensa recibida $k$ pasos de tiempo en el futuro vale sólo $\gamma^{k-1}$ veces lo que valdría si se recibiera inmediatamente. Si $\gamma < 1$, la suma infinita en (3.7) tiene un valor finito mientras la secuencia de recompensas $\{R_k\}$ esté acotada. A medida que $\gamma$ se acerca a 1, el objetivo de rendimiento tiene más en cuenta las recompensas futuras.

Los retornos en los sucesivos pasos de tiempo están relacionados entre sí de una manera que es importante para la teoría y los algoritmos de aprendizaje por refuerzo:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{3.8}$$

Cabe destacar que esto funciona para todos los pasos de tiempo $t < T$, incluso si la terminación se produce en $t + 1$, si definimos $G_T = 0$.

Nótese que aunque el retorno (3.7) es una suma de un número infinito de términos, sigue siendo finito si la recompensa es distinta de cero y constante, si $\gamma < 1$. Por ejemplo, si la recompensa es una constante +1, entonces el retorno es

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}. \tag{3.9}$$

También podemos definir el retorno de forma que funcione tanto para tareas episódicas como continuas:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k. \tag{3.10}$$

## 3.5. Políticas y Funciones de Valor

Casi todos los algoritmos de aprendizaje por refuerzo implican la estimación de funciones de valor, funciones de estados (o de pares estado-acción) que estiman *cómo de bueno* es para el agente estar en un estado dado (o cómo de bueno es realizar una acción dada en un estado dado). En este contexto, la noción de "cómo de bueno" se define en términos de las recompensas futuras que se pueden esperar o, para ser precisos, en términos del retorno esperado. En consecuencia, las funciones de valor se definen con respecto a formas particulares de actuar, llamadas políticas.

Formalmente, una *política* es un mapeo de estados a probabilidades de seleccionar cada acción posible. Si el agente sigue la política $\pi$ en el momento $t$, entonces $\pi(a|s)$ es la probabilidad de que $A_t = a$ si $S_t = s$. Un caso especial es el de las *políticas deterministas*, en el que denotamos la acción que se seleccionará en el estado $s$ como $\pi(s)$. Los métodos de aprendizaje por refuerzo especifican cómo cambia la política del agente como resultado de su experiencia.

La *función de valor* de un estado $s$ bajo una política $\pi$, denotada $v_\pi(s)$, es el retorno esperado al comenzar en $s$ y seguir $\pi$ a partir de entonces. Llamamos a la función $v_\pi$ la *función de valor del estado para la política* $\pi$. Para los MDP, podemos definir $v_\pi$ formalmente por

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s \right], \tag{3.11}$$

Del mismo modo, definimos el valor de tomar la acción $a$ en el estado $s$ bajo una política $\pi$, denotado $q_\pi(s, a)$, como el retorno esperado al comenzar desde $s$, tomar la acción $a$ y luego seguir la política $\pi$:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a \right]. \tag{3.12}$$

Llamamos a $q_\pi$ la *función de valor de acción para la política* $\pi$.

Mientras que $v_\pi(s)$ solo considera la calidad de los estados, $q_\pi(s, a)$ nos permite evaluar y comparar la calidad de las diferentes acciones posibles en un estado dado, proporcionando una herramienta más detallada para la toma de decisiones en un entorno de aprendizaje por refuerzo.

Una propiedad fundamental de las funciones de valor es que satisfacen relaciones recursivas similares a las del retorno (3.8). Para cualquier política $\pi$ y cualquier estado $s$, se cumple la siguiente condición de consistencia entre el valor de $s$ y el valor de sus posibles estados sucesores:

$$\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] && \text{(por (3.8))} \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[ r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s'] \right] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) \left[ r + \gamma v_\pi(s') \right]. && (3.13)
\end{aligned}$$

Nótese cómo la expresión final se puede leer fácilmente como un valor esperado. Es una suma sobre todas las posibles combinaciones de las tres variables: $a$, $s'$ y $r$. Para cada triple, calculamos su probabilidad, $\pi(a|s)p(s', r \mid s, a)$, ponderamos la cantidad entre corchetes por esa probabilidad, y luego sumamos todos estos productos para obtener un valor esperado.

La Ecuación (3.13) es la ecuación de Bellman para $v_\pi$. Expresa una relación entre el valor de un estado y los valores de sus estados sucesores. Piense en mirar hacia adelante desde un estado hasta sus posibles estados sucesores.

La derivación de la ecuación de Bellman para $q_\pi(s,a)$ se deriva de forma natural de nuestra comprensión de la función de valor del estado, pero abre poderosas posibilidades para procesos de toma de decisiones más detallados y estratégicos al capturar el valor de tomar acciones específicas en estados específicos. La ecuación de Bellman para $q_\pi(s,a)$ se puede expresar como:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\right] \\ &= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right] \\ &= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(a', s')\right]. \end{aligned} \tag{3.14}$$

## 3.6. Políticas Óptimas y Funciones de Valor Óptimas

Resolver una tarea de aprendizaje por refuerzo significa, en términos generales, encontrar una política que logre una gran recompensa a largo plazo. Para los MDP finitos, podemos definir con precisión una política óptima de la siguiente manera. Una política $\pi$ se define como mejor que o igual a una política $\pi'$ si su retorno esperado es mayor o igual que el de $\pi'$ para todos los estados. En otras palabras, $\pi \geq \pi'$ si y solo si $v_\pi(s) \geq v_{\pi'}(s)$ para todos $s \in \mathcal{S}$. Siempre existe al menos una política que es mejor o igual a todas las demás políticas. Esta es una *política óptima*. Aunque puede haber más de una, denotamos todas las políticas óptimas por $\pi_*$. Todas ellas comparten la misma función de valor del estado, llamada *función de valor óptima del estado*, denotada $v_*$, y definida como

$$v_*(s) \doteq \max_\pi v_\pi(s), \tag{3.15}$$

para todo $s \in \mathcal{S}$. Las políticas óptimas también comparten la misma *función de valor óptima de la acción*, definida como

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \tag{3.16}$$

para todo $s \in \mathcal{S}$ y $a \in \mathcal{A}(s)$. Esta función proporciona el retorno esperado por tomar la acción $a$ en el estado $s$ y después seguir una política óptima. Así, podemos escribir $q_*$ en términos de $v_*$ de la siguiente manera:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]. \tag{3.17}$$

Esto nos lleva a un concepto fundamental en el aprendizaje por refuerzo: la ecuación de Bellman para $v_*$, o la *ecuación de optimalidad de Bellman*. Intuitivamente, la ecuación de optimalidad de Bellman expresa el hecho de que el valor de un estado bajo una política óptima debe ser igual al retorno esperado para la mejor acción desde ese estado:

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] && \text{(por (3.8))} \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] && (3.18) \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_*(s')\right]. && (3.19) \end{aligned}$$

Las dos últimas ecuaciones son dos formas de la ecuación de optimalidad de Bellman para $v_*$. La ecuación de optimalidad de Bellman para $q_*$ es

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]. && (3.20) \end{aligned}$$

## 3.7. Resolución de Procesos de Decisión de Markov Finitos

Aunque es teóricamente factible resolver las ecuaciones de Bellman para todos los MDP finitos cuando se conoce la función dinámica $p$, la complejidad computacional se vuelve intratable a medida que el tamaño del MDP aumenta significativamente.

Dada la función de valor óptima, determinar la política óptima es sencillo. Si se conoce la función de valor del estado óptima $v_*(s)$, la política óptima se puede derivar simplemente mirando un paso adelante en cada estado y seleccionando la acción que maximiza las recompensas esperadas. Formalmente, esto se puede expresar mediante la selección de acciones de acuerdo con la siguiente ecuación:

$$\pi_*(s) = \arg\max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_*(s')\right], \tag{3.21}$$

Si se conoce la función de valor de acción óptima $q_*(s,a)$, la determinación de la política óptima resulta aún más sencilla. En cada estado, la acción óptima puede identificarse seleccionando la acción con la función de valor de acción más alta:

$$\pi_*(s) = \arg\max_{a \in \mathcal{A}(s)} q_*(s,a). \tag{3.22}$$

CAPÍTULO 4

# Programación Dinámica

La Programación Dinámica (PD) se refiere a una colección de algoritmos que se emplean para calcular políticas óptimas cuando tenemos un modelo perfecto del entorno representado como un Proceso de Decisión de Markov (*Markov Decision Process*, o MDP).

Aunque los algoritmos clásicos de la PD tienen una utilidad limitada en el aprendizaje por refuerzo debido a sus suposiciones de un modelo perfecto y los considerables recursos computacionales que requieren, son fundamentales para la comprensión teórica, ya que proporcionan la base para muchos métodos que buscan lograr resultados similares pero con menor computación y sin asumir un modelo perfecto del entorno.

Normalmente, suponemos que el entorno es un MDP finito. Es decir, asumimos que sus conjuntos de estados, acciones y recompensas, $\mathcal{S}$, $\mathcal{A}$ y $\mathcal{R}$, son finitos, y que su dinámica está dada por un conjunto de probabilidades $p(s', r \mid s, a)$, para todo $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $r \in \mathcal{R}$ y $s' \in \mathcal{S}^+$ ($\mathcal{S}^+$ es $\mathcal{S}$ más un estado terminal si el problema es episódico).

La idea central de la PD, y del aprendizaje por refuerzo en general, es el uso de funciones de valor para organizar y guiar la búsqueda de políticas óptimas. Como se ha comentado en la Sección 3, podemos obtener fácilmente políticas óptimas una vez que hemos encontrado las funciones de valor óptimas, $v_*$ o $q_*$, que satisfacen las ecuaciones de optimalidad de Bellman, (3.19) y (3.20).

## 4.1. Evaluación de Políticas

En primer lugar examinaremos cómo calcular la función de valor de estado $v_\pi$ para una política arbitraria $\pi$. Esto se denomina *evaluación de políticas* en la literatura de la PD. También nos referimos a esto como el *problema de predicción*.

Recordemos de la Sección 3 que, para todo $s \in \mathcal{S}$,

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] && \text{(por (3.8))} \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right]. && (3.13) \end{aligned}$$

donde $\pi(a|s)$ es la probabilidad de tomar la acción $a$ en el estado $s$ bajo la política $\pi$, y las expectativas están subscritas por $\pi$ para indicar que están condicionadas a que se siga $\pi$.

Si la dinámica del entorno es completamente conocida, entonces el problema de evaluación de políticas puede formularse como un sistema de $|\mathcal{S}|$ ecuaciones lineales con $|\mathcal{S}|$ incógnitas (los $v_\pi(s), s \in \mathcal{S}$). Estas ecuaciones lineales son las ecuaciones de Bellman para cada estado, como se da en la ecuación (3.13). Sin embargo, a medida que aumenta el tamaño del MDP, la complejidad computacional de resolver este sistema de ecuaciones crece rápidamente. En consecuencia, en la práctica, se emplean métodos iterativos para calcular la función de valor.

El proceso implica la creación de una secuencia de funciones de valor aproximadas $v_0, v_1, v_2, \ldots$, cada una de las cuales mapea $\mathcal{S}^+ \to \mathbb{R}$, que convergen a la verdadera función de valor $v_\pi$ a medida que se realizan más iteraciones.

La aproximación inicial, $v_0$, se elige arbitrariamente (excepto para el estado terminal, que si lo hay, se le debe dar valor 0), y cada aproximación sucesiva se obtiene utilizando la ecuación de Bellman para $v_\pi$ (3.13) como regla de actualización:

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s')\right], && (4.1) \end{aligned}$$

para todo $s \in \mathcal{S}$. Aplicada iterativamente, esta regla de actualización hará que la secuencia $\{v_k\}$ converja a $v_\pi$ a medida que $k \to \infty$ bajo ciertas condiciones, como cuando el factor de descuento $\gamma$ es menor que 1, o si la política asegura la terminación eventual desde todos los estados. Este algoritmo se llama *evaluación iterativa de políticas*.

En la práctica, este algoritmo puede implementarse utilizando dos arrays (uno para los valores antiguos y otro para los nuevos) o un único array donde las actualizaciones se realizan *in situ*. El método *in situ* generalmente converge más rápido, ya que utiliza la nueva información tan pronto como está disponible.

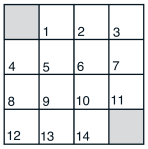
A continuación, se muestra en pseudocódigo una versión completa de la evaluación iterativa de políticas *in situ*. Obsérvese cómo se gestiona la terminación. Formalmente, la evaluación iterativa de políticas solo converge en el límite, pero en la práctica debe detenerse antes de esto. El pseudocódigo prueba la cantidad $\max_{s \in \mathcal{S}} |v_{k+1}(s) - v_k(s)|$ después de cada ciclo y se detiene cuando es suficientemente pequeño.

**Algorithm 1** Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input $\pi$, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$
Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

### 4.1.1. Ejemplo Evaluación de Políticas: Cuadrículas

Considere las cuadrículas de $4 \times 4$ mostradas a continuación.




Los estados no terminales son $S = \{1, 2, \ldots, 14\}$. Hay cuatro acciones posibles en cada estado, $A = \{\text{arriba, abajo, derecha, izquierda}\}$, que determinan de manera determinista las transiciones de estado correspondientes, excepto en el caso de las acciones que sacarían al agente de las cuadrículas, en cuyo caso el estado permanece sin cambios. La tarea es episódica y sin descuento, con una recompensa de $-1$ en todas las transiciones hasta que se alcanza el estado terminal. El estado terminal está sombreado en la figura (aunque se muestra en dos lugares, formalmente es un solo estado).

Supongamos que el agente sigue la política aleatoria equiprobable (todas las acciones igual de probables). El lado izquierdo de la Figura 4.1 muestra la secuencia de funciones de valor $\{v_k\}$ calculadas mediante la evaluación iterativa de políticas. La estimación final es, de hecho, $v_\pi$, que en este caso indica el número esperado de pasos desde ese estado hasta la terminación en negativo. La columna derecha es la secuencia de políticas codiciosas correspondientes a las estimaciones de la función de valor. En este caso, todas las políticas después de la tercera iteración son óptimas.

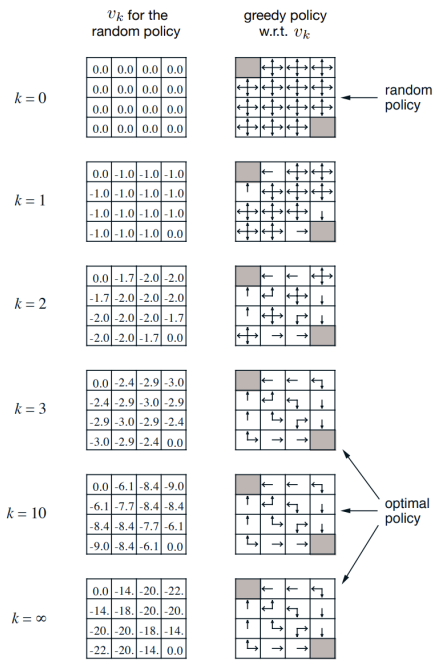



**Figura 4.1:** Convergencia de la evaluación iterativa de políticas en unas cuadrículas pequeñas

## 4.2. Mejora de Políticas

La razón para calcular la función de valor para una política es ayudar a encontrar mejores políticas. Supongamos que hemos determinado la función de valor $v_\pi$ para una política determinista arbitraria, $\pi$. Para decidir si deberíamos cambiar la política en el estado $s$ para elegir una acción $a \neq \pi(s)$, comparamos $v_\pi(s)$ con el valor de tomar la acción $a$ en $s$ y, a partir de ahí, seguir $\pi$, dado por:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right] \end{aligned} \tag{4.2}$$

Si $q_\pi(s,a)$ es mayor que $v_\pi(s)$, sugiere que la elección sistemática de $a$ en $s$ llevaría a una política mejor en términos generales.

Esta idea es un caso específico del más amplio *teorema de mejora de políticas*. Según este teorema, si $\pi$ y $\pi'$ son cualquier par de políticas deterministas tales que, para todo $s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s), \tag{4.3}$$

entonces la política $\pi'$ debe ser mejor o igual de buena que $\pi$. Es decir, debe obtener un retorno esperado mayor o igual desde todos los estados $s \in \mathcal{S}$:

$$v_{\pi'}(s) \geq v_\pi(s). \tag{4.4}$$

De hecho, si hay una desigualdad estricta en (4.3) en algún estado, entonces debe haber una desigualdad estricta en (4.4) en ese estado.

El teorema de mejora de políticas se aplica a las dos políticas que consideramos al principio de esta sección, una política determinista original: $\pi$, y una política cambiada, $\pi'$, que es idéntica a $\pi$ excepto que $\pi'(s) = a \neq \pi(s)$. Para los estados distintos de $s$, (4.3) se cumple porque ambos lados son iguales. Por lo tanto, si $q_\pi(s,a) > v_\pi(s)$, entonces la política cambiada es mejor que $\pi$.

Es natural extrapolar esta idea para considerar cambios en *todos* los estados y hacia *todas* las posibles acciones, seleccionando en cada estado la acción que parezca mejor según $q_\pi(s,a)$. En otras palabras, considerar la nueva *política codiciosa*, $\pi'$, dada por

$$\begin{aligned} \pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right], \end{aligned} \tag{4.5}$$

donde $\arg\max_a$ denota el valor de $a$ en el que se maximiza la expresión que sigue (con los empates resueltos de manera arbitraria).

Por su propia naturaleza, la política codiciosa cumple las condiciones del teorema de mejora de políticas (4.3), por lo que sabemos que es tan buena o mejor que la política original. El proceso de crear una nueva política que mejora sobre una política original, haciéndola codiciosa con respecto a la función de valor de la política original, se llama *mejora de políticas*.

Supongamos que la nueva política codiciosa, $\pi'$, es tan buena como, pero no mejor que, la antigua política $\pi$. Entonces $v_{\pi'} = v_\pi$, y a partir de (4.5) se deduce que para todo $s \in \mathcal{S}$:

$$\begin{aligned} v_{\pi'}(s) &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_{\pi'}(s')\right]. \end{aligned}$$

Pero esto coincide con la ecuación de optimalidad de Bellman (3.19), por lo tanto, $v_{\pi'}$ debe ser $v_*$. De este modo, tanto $\pi$ como $\pi'$ deben ser políticas óptimas. En consecuencia, la mejora de políticas nos debe dar una política estrictamente mejor, excepto cuando la política original es la óptima.

Todas las ideas de esta sección se pueden extrapolar fácilmente a las políticas estocásticas. Una *política estocástica* $\pi$ especifica probabilidades, $\pi(a \mid s)$, para tomar cada acción, $a$, en cada estado, $s$. Si hay empates en los pasos de mejora de políticas, la probabilidad puede distribuirse entre todas las acciones que maximizadoras, se permite cualquier esquema de distribución, siempre que todas las acciones submáximas se les asigne una probabilidad de cero.

## 4.3. Iteración de Políticas

Una vez que una política, $\pi$, ha sido mejorada utilizando $v_\pi$ para obtener una mejor política, $\pi'$, podemos entonces calcular $v_{\pi'}$ y mejorarla nuevamente para obtener una política aún mejor, $\pi''$. De esta manera, podemos obtener una secuencia de políticas y funciones de valor que mejoran monotónicamente:

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

donde $\xrightarrow{\text{E}}$ denota una *evaluación de políticas (policy evaluation)* y $\xrightarrow{\text{I}}$ denota una *mejora de políticas (policy improvement)*. Se garantiza que cada política es una mejora estricta sobre la anterior (a menos que ya sea óptima).

Esta forma de encontrar una política óptima se llama *iteración de políticas*. A continuación se presenta un algoritmo completo. Nótese que cada evaluación de política, que es un cálculo iterativo, se inicia con la función de valor de la política anterior. Esto típicamente resulta en un gran aumento en la velocidad de convergencia de la evaluación de políticas (presumiblemente porque la función de valor cambia poco de una política a la siguiente).

**Algorithm 2** Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s')\right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (A small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   Loop for each $s \in \mathcal{S}$:
      *old-action* $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s')\right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

## 4.4. Iteración de Valores

Una desventaja de la iteración de políticas es que cada iteración requiere una evaluación de políticas, que puede ser un proceso iterativo extenso que involucra múltiples barridos a través del conjunto de estados. Cuando se utilizan métodos iterativos, la convergencia exacta a $v_\pi$ ocurre solo en el límite. Sin embargo, no siempre es necesario esperar a la convergencia total. Tal como se ilustra en la Figura 4.1, truncar el proceso de evaluación de políticas puede ser suficiente, ya que las iteraciones adicionales más allá de cierto punto, a menudo, no afectan la política codiciosa resultante.

Esto nos conduce al concepto del algoritmo de *iteración de valores*, que detiene la evaluación de políticas después de solo un barrido (una actualización de cada estado). Se puede escribir como una operación de actualización particularmente simple que combina los pasos de mejora de políticas y evaluación de políticas truncada:

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s')\right], \tag{4.6}$$

para todo $s \in S$.

Nótese que la iteración de valores se obtiene simplemente convirtiendo la ecuación de optimalidad de Bellman (3.19) en una regla de actualización. Obsérvese también cómo la actualización de la iteración de valores es idéntica a la actualización de la evaluación de políticas (4.1), excepto que requiere tomar el máximo sobre todas las acciones.

**Algorithm 3** Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$
Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$
Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma V(s')\right]$

## 4.5. Iteración de Políticas Generalizada

El término *iteración de políticas generalizada* (GPI, *Generalized Policy Iteration*) se refiere a la idea general de permitir que los procesos de evaluación de políticas y mejora de políticas interactúen, independientemente de la granularidad y otros detalles de los dos procesos. Casi todos los métodos de aprendizaje por refuerzo se describen bien como GPI. Es decir, todos tienen políticas y funciones de valor identificables, donde la política siempre se mejora con respecto a la función de valor y la función de valor siempre se dirige hacia la función de valor para la política, como se sugiere en la Figura 4.2.

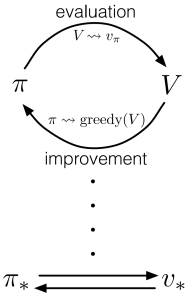



**Figura 4.2:** *Iteración de Políticas Generalizada* (GPI)

Los procesos de evaluación y mejora en GPI compiten y cooperan. Sin embargo, trabajan juntos para encontrar una solución conjunta, en el largo plazo: la función de valor y la política óptimas.

CAPÍTULO 5

# Métodos de Monte Carlo

En esta sección, exploramos los métodos de Monte Carlo como un primer enfoque para estimar funciones de valor y descubrir políticas óptimas en el aprendizaje por refuerzo, sin asumir un conocimiento completo del entorno. A diferencia de la Programación Dinámica, donde *calculamos* las funciones de valor a partir del conocimiento del MDP, los métodos de Monte Carlo solo requieren experiencia, es decir, *aprenden* las funciones de valor a partir de retornos de muestra obtenidos a través de interacciones reales o simuladas con el MDP.

Los métodos de Monte Carlo se centran en promediar los retornos de las muestras para resolver el problema de aprendizaje por refuerzo. Consideramos estos métodos específicamente para tareas episódicas, donde la experiencia se divide en episodios que eventualmente terminan. Las estimaciones de valor y las políticas se actualizan solo después de la finalización de un episodio, lo que hace que estos métodos sean incrementales episodio a episodio, pero no paso a paso.

## 5.1. Predicción Monte Carlo

Comenzamos considerando los métodos de Monte Carlo para aprender la función de valor de estado para una política determinada. Recordemos que el valor de un estado es el retorno esperado, la recompensa futura acumulativa descontada esperada, comenzando desde ese estado. Una forma obvia de estimarlo a partir de la experiencia es simplemente promediando los retornos observados después de las visitas a ese estado. A medida que se observan más retornos, el promedio debería converger al valor esperado. Esta idea subyace en todos los métodos de Monte Carlo.

En concreto, supongamos que deseamos estimar $v_\pi(s)$, el valor de un estado $s$ bajo la política $\pi$, dado un conjunto de episodios obtenidos siguiendo $\pi$ y pasando por $s$. Cada aparición del estado $s$ en un episodio se llama una *visita* a $s$. Por supuesto, $s$ puede ser visitado varias veces en el mismo episodio; llamemos a la primera vez que se visita en un episodio la *primera visita* a $s$.

Existen dos métodos principales en la Predicción Monte Carlo:

1. *First-Visit Monte Carlo*: Este método estima $v_\pi(s)$ como el promedio de los retornos después de las primeras visitas a $s$ en cada episodio.

2. *Every-Visit Monte Carlo*: Este método, por otro lado, promedia los retornos después de todas las visitas a $s$.

Aunque son ligeramente diferentes en sus propiedades teóricas, ambos métodos finalmente convergen a $v_\pi(s)$ a medida que el número de visitas (o primeras visitas) a $s$ tiende a infinito.

## 5.2. Estimación Monte Carlo de Valores de Acción

Si no se dispone de un modelo, entonces es particularmente útil estimar los valores de *acción* (los valores de los pares estado-acción) en lugar de los valores de *estado*. Con un modelo, los valores de estado por sí solos son suficientes para determinar una política; uno simplemente observa un paso hacia adelante y elige la acción que conduce a la mejor combinación de recompensa y el siguiente estado. Sin embargo, sin un modelo, los valores de estado por sí solos no son suficientes. Por lo tanto, uno de nuestros objetivos principales para los métodos de Monte Carlo es estimar $q_*$.

Luego, el problema de la evaluación de políticas para los valores de acción es estimar $q_\pi(s, a)$, el retorno esperado al comenzar en el estado $s$, tomar la acción $a$, y seguir la política $\pi$ a partir de entonces. Los métodos de Monte Carlo para esto son esencialmente los mismos que los empleados para los valores de estado, excepto que ahora hablamos de visitas a un par estado-acción en lugar de a un estado. Se dice que un par estado-acción $(s, a)$ ha sido visitado en un episodio si alguna vez se visita el estado $s$ y se toma la acción $a$ en él.

La única complicación es que muchos pares estado-acción pueden no ser visitados nunca. Este es el problema general de *mantener la exploración*. Una forma de hacer esto es especificar que los episodios *comienzan en un par estado-acción*, y que cada par tiene una probabilidad distinta de cero de ser seleccionado como inicio. Esto garantiza que todos los pares estado-acción serán visitados un número infinito de veces en el límite de un número infinito de episodios. A esto se le llama la suposición de *inicios exploratorios*. La suposición de inicios exploratorios a veces es útil, pero no siempre es aplicable, como al aprender directamente de la interacción real con un entorno. El enfoque alternativo más común para asegurar que todos los pares estado-acción sean encontrados es considerar solo políticas que sean estocásticas, es decir, políticas con una probabilidad distinta de cero de seleccionar cualquiera de las acciones en cada estado.

## 5.3. Control de Monte Carlo

Ahora estamos listos para considerar cómo la estimación Monte Carlo puede utilizarse en el control, es decir, para aproximar políticas óptimas. La idea general es proceder según el concepto de iteración de políticas generalizada (GPI). La *evaluación de políticas* se realiza experimentando muchos episodios, con la función de valor de acción aproximada acercándose asintóticamente a la función verdadera. La *mejora de políticas* se lleva a cabo haciendo que la política sea codiciosa con respecto a la función de valor de acción actual. El proceso de alternar entre estos dos pasos continúa hasta que la política converge a la política óptima.

Para garantizar fácilmente la convergencia del método de Monte Carlo, debemos partir de dos premisas fundamentales. Una es que los episodios tienen inicios exploratorios, y la otra es que la evaluación de políticas se puede realizar con un número infinito de episodios. Pero, para obtener un algoritmo práctico, tendremos que eliminar ambas premisas. Un algoritmo completo y simple en esta línea, que llamamos *Monte Carlo ES* (Monte Carlo con Inicios Exploratorios), se presenta en pseudocódigo en el Algoritmo 4.

**Algorithm 4** Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  $N(s, a) \leftarrow 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Loop forever (for each episode) :
  Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability $> 0$
  Generate an episode from $(S_0, A_0)$, following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    If $(S_t, A_t) \notin \{(S_0, A_0), (S_1, A_1), \ldots, (S_{t-1}, A_{t-1})\}$, then
      $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} [G - Q(S_t, A_t)]$
      $\pi(S_t) \leftarrow \arg \max_a Q(S_t, a)$

## 5.4. Monte Carlo On-policy

Para evitar la poco realista suposición de inicios exploratorios, necesitamos asegurar que todas las acciones se seleccionen un número infinito de veces. Hay dos enfoques para lograr esto, lo que nos lleva a lo que llamamos métodos *on-policy* y métodos *off-policy*.

Los métodos *on-policy* intentan evaluar o mejorar la misma política que el agente utiliza para tomar decisiones durante el aprendizaje. Es decir, la política que el agente sigue para generar los datos es la misma que se intenta optimizar. Por otro lado, los métodos *off-policy* evalúan o mejoran una política diferente de la que se utiliza para generar los datos. En este caso, el agente puede seguir una política exploratoria para recopilar datos mientras optimiza o evalúa otra política más centrada en la explotación de la información aprendida.

En los métodos de control on-policy, la política generalmente es suave (*soft*), lo que significa que $\pi(a|s) > 0$ para todo $s \in \mathcal{S}$ y todo $a \in \mathcal{A}(s)$, pero se ajusta gradualmente para acercarse cada vez más a una política óptima determinista. El método on-policy que presentamos utiliza políticas *$\varepsilon$-greedy*. Es decir, a todas las acciones no codiciosas se les asigna la probabilidad mínima de selección, $\frac{\varepsilon}{|\mathcal{A}(s)|}$, y el resto de la probabilidad, $1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|}$, se asigna a la acción codiciosa. Las políticas *$\varepsilon$-greedy* son ejemplos de políticas *$\varepsilon$-soft*, definidas como políticas para las cuales $\pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$ para todos los estados y acciones, para algún $\varepsilon > 0$. Entre las políticas *$\varepsilon$-soft*, las políticas *$\varepsilon$-greedy* son, en cierto sentido, las más cercanas a las políticas codiciosas. El algoritmo completo se presenta a continuación.

**Algorithm 5** On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$
Initialize:
  $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
  $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Loop forever (for each episode) :
  Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$:
      Append $G$ to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow$ average($Returns(S_t, A_t)$)
      $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(S_t)|} & \text{if } a = A^* \\ \frac{\varepsilon}{|\mathcal{A}(S_t)|} & \text{if } a \neq A^* \end{cases}$$

## 5.5. Predicción Off-policy via Importance Sampling

Todos los métodos de control de aprendizaje enfrentan un dilema: buscan aprender los valores de acción para un comportamiento óptimo, pero deben actuar de manera no óptima para explorar todas las acciones. El enfoque on-policy es un compromiso, ya que aprende los valores de acción para una política casi óptima que aún explora. Un enfoque más directo es utilizar dos políticas: la *política objetivo* o *target policy*, que se está aprendiendo y se convierte en óptima, y la *política de comportamiento* o *behavior policy*, que es más exploratoria y genera el comportamiento. Este proceso se conoce como *aprendizaje off-policy*.

Los métodos off-policy se basan en la idea del *importance sampling*, una técnica utilizada para estimar valores esperados bajo una distribución de probabilidad mientras se muestrea de otra. El *importance sampling* nos permite ajustar los retornos obtenidos bajo la política de comportamiento para estimar cuáles habrían sido los retornos obtenidos bajo la política objetivo.

Aplicamos el *importance sampling* al aprendizaje off-policy ponderando los retornos según la probabilidad relativa de que sus trayectorias ocurran bajo las políticas objetivo y de comportamiento, lo que se denomina el ratio de muestreo de importancia, o *importance-sampling ratio*.

Dado un estado inicial $S_t$, la probabilidad de que ocurra la trayectoria subsecuente estado-acción,
$A_t, S_{t+1}, A_{t+1}, \dots, S_T$, bajo cualquier política $\pi$ es

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \dots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t \mid S_t) p(S_{t+1} \mid S_t, A_t) \pi(A_{t+1} \mid S_{t+1}) \cdots p(S_T \mid S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k),
\end{aligned}$$

donde $p$ es la función de probabilidad de transición de estados definida por (3.2). Así, la probabilidad relativa de la trayectoria bajo las políticas objetivo y de comportamiento (*importance-sampling ratio*) es

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}. \tag{5.1}$$

Aunque las probabilidades de la trayectoria dependen de las probabilidades de transición del MDPque generalmente se desconocen, estas aparecen de manera idéntica tanto en el numerador como en el denominador, y por lo tanto se cancelan. El cociente de muestreo de importancia termina dependiendo solo de las dos políticas y de la secuencia, no del MDP.

La expectativa original $\mathbb{E}[G_t \mid S_t = s] = v_b(s)$ no produce $v_\pi$, pero el *importance-sampling ratio* $\rho_{t:T-1}$ corrige esto:

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s). \tag{5.2}$$

Existen dos tipos principales de *importance sampling* utilizados en el aprendizaje off-policy: *importance sampling ordinario* y *importance sampling ponderado*.

El *importance sampling ordinario* implica escalar los retornos mediante el ratio de *importance sampling* y luego promediarlos. Aunque es insesgado, lo que significa que convergerá al valor correcto a medida que aumente el número de muestras, tiene el inconveniente de tener potencialmente una varianza alta o incluso infinita

Por otro lado, el *importance sampling ponderado*, normaliza estos ratios, lo que reduce la varianza pero introduce un pequeño sesgo. Generalmente se prefiere en la práctica debido a sus propiedades de convergencia más estables.

Para estimar $q_\pi(s,a)$ es conveniente numerar los pasos de tiempo de forma que aumenten a través de los límites de los episodios. Es decir, si el primer episodio del conjunto termina en el tiempo 100, entonces el siguiente episodio comienza en el tiempo $t = 101$. Sea $\mathcal{T}(s,a)$ el conjunto de pasos de tiempo donde el par de estado-acción $(s,a)$ fue visitado por primera vez si se utiliza el método de primera visita, o el conjunto de pasos de tiempo para todas las visitas si se utiliza el método de todas las visitas. Además, sea $T(t)$ el primer momento de la terminación después del tiempo $t$, y $G_t$ el retorno después de $t$ hasta $T(t)$.

Usando el *importance sampling ordinario*:

$$Q(s,a) \doteq \frac{\sum_{t\in\mathcal{T}(s,a)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s,a)|}$$

Usando el *importance sampling ponderado*:

$$Q(s,a) \doteq \frac{\sum_{t\in\mathcal{T}(s,a)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\mathcal{T}(s,a)} \rho_{t:T(t)-1}},$$

A continuación se presenta un algoritmo MC de todas las visitas (*every-visit*) para la evaluación de políticas off-policy utilizando muestreo de importancia ponderado.

**Algorithm 6** Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
- $Q(s,a) \in \mathbb{R}$ (arbitrarily)
- $C(s,a) \leftarrow 0$
- $\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode) :
- $b \leftarrow$ any soft policy
- Generate an episode using $b$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- $W \leftarrow 1$
- Loop for each for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)}[G - Q(S_t, A_t)]$
  - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
  - If $A_t \neq \pi(S_t)$, then exit inner Loop (proceed to next episode)
  - $W \leftarrow W \frac{1}{b(A_t|S_t)}$

CAPÍTULO 6

# Aprendizaje por Diferencia Temporal

Si hubiera que identificar una idea como central y novedosa en el aprendizaje por refuerzo, sin duda sería el *aprendizaje por diferencia temporal* (TD, *Temporal-Difference*). El aprendizaje TD destaca porque cierra la brecha entre los métodos de Monte Carlo y la Programación Dinámica. Al igual que los métodos de Monte Carlo, los métodos TD pueden aprender directamente de la experiencia sin requerir un modelo de la dinámica del entorno. Sin embargo, a diferencia de Monte Carlo, que espera hasta el final de un episodio para actualizar sus estimaciones, el aprendizaje TD actualiza sus estimaciones en cada paso de tiempo, combinando las recompensas observadas con sus estimaciones actuales. Este proceso de actualización inmediata se conoce como "bootstrapping", una característica compartida con la Programación Dinámica.

La relación entre los métodos TD, PD y Monte Carlo es un tema recurrente en la teoría del aprendizaje por refuerzo.

## 6.1. Predicción TD

La predicción TD se ocupa del problema de evaluación de políticas, que implica estimar la función de valor $v_\pi$ para una política dada $\pi$. El método TD más simple, conocido como TD(0), actualiza la estimación de la función de valor $V$ para un estado $S_t$ inmediatamente después de la transición al siguiente estado $S_{t+1}$, utilizando la recompensa observada $R_{t+1}$ y la estimación $V(S_{t+1})$. La regla de actualización es:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right] \tag{6.1}$$

Aquí, $\alpha$ es un parámetro de tamaño de paso, y $\gamma$ es el factor de descuento. Este método se denomina TD(0) porque basa sus actualizaciones en una anticipación de un solo paso, lo que lo convierte en la forma más simple de aprendizaje TD. El siguiente algoritmo especifica TD(0) completamente en forma de procedimiento.

**Algorithm 7** Tabular TD(0) for estimating $v_\pi$

Input: the policy $\pi$ to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in S^+$, arbitrarily except that $V(terminal) = 0$
Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        $A \leftarrow$ action given by $\pi$ for $S$
        Take action $A$, observe $R$, $S'$
        $V(S) \leftarrow V(S) + \alpha \left[R + \gamma V(S') - V(S)\right]$
        $S \leftarrow S'$
    until $S$ is terminal

## 6.2. Ventajas del Aprendizaje TD

Los métodos TD tienen una ventaja sobre los métodos de PD pues no requieren un modelo del entorno, de su recompensa y de las distribuciones de probabilidad del siguiente estado. Los métodos TD pueden aprender directamente de la experiencia cruda, lo que los hace aplicables a una gama más amplia de problemas.

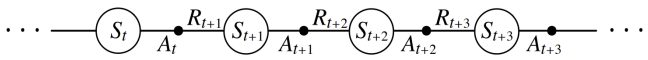
La siguiente ventaja más evidente de los métodos TD sobre los métodos de Monte Carlo es que se implementan de manera natural en un modo online, completamente incremental. Los métodos TD actualizan sus estimaciones después de cada paso de tiempo, lo que permite un aprendizaje en tiempo real. Esto es particularmente ventajoso en entornos donde los episodios son largos o donde no hay límites claros entre episodios.

Tanto los métodos TD como los de Monte Carlo convergen asintóticamente a las predicciones correctas, por lo que una pregunta natural es ¿qué método aprende más rápido? En la actualidad, esta es una pregunta abierta en el sentido de que nadie ha podido demostrar matemáticamente que un método converge más rápido que el otro. Sin embargo, en la práctica, se ha encontrado generalmente que los métodos TD tienden a converger más rápido que los métodos de Monte Carlo en tareas estocásticas.

## 6.3. Sarsa: Control TD On-policy

En esta sección presentamos un método de control TD on-policy.

El primer paso es aprender una función de valor de acción en lugar de una función de valor de estado. En particular, para un método en on-policy debemos estimar $q_\pi(s, a)$ para la política de comportamiento actual $\pi$ y para todos los estados $s$ y acciones $a$. Esto se puede hacer utilizando esencialmente el mismo método TD descrito anteriormente para aprender $v_\pi$. Recordemos que un episodio consiste en una secuencia alternada de estados y pares estado-acción:
El nombre Sarsa proviene del quíntuplo de variables que representan los pasos en el




proceso de aprendizaje: el estado actual $S_t$, la acción tomada $A_t$, la recompensa recibida $R_{t+1}$, el siguiente estado $S_{t+1}$, y la acción tomada en el siguiente estado $A_{t+1}$. La regla de actualización para Sarsa es

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]. \tag{6.2}$$

Resulta sencillo diseñar un algoritmo de control on-policy basado en el método de predicción Sarsa. Como en todos los métodos on-policy, estimamos continuamente $q_\pi$ para la política de comportamiento $\pi$, y al mismo tiempo cambiamos $\pi$ hacia la voracidad con respecto a $q_\pi$. La forma general del algoritmo de control Sarsa se presenta en el Algoritmo 8.

**Algorithm 8** Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$
Loop for each episode:
    Initialize $S$
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Loop for each step of episode:
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma Q(S', A') - Q(S, A)\right]$
        $S \leftarrow S'$; $A \leftarrow A'$
until $S$ is terminal

## 6.4. Q-learning: Control TD Off-policy

Uno de los primeros avances en el aprendizaje por refuerzo fue el desarrollo de un algoritmo de control TD off-policy conocido como *Q-learning*, definido por

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{6.3}$$

En este caso, la función de valor de acción aprendida, $Q$, se aproxima directamente a $q_*$, la función de valor de acción óptima, independientemente de la política que se esté siguiendo. Esto simplifica drásticamente el análisis del algoritmo y permitió las primeras pruebas de convergencia.

A continuación se presenta el pseudocódigo para Q-learning, donde la política objetivo es la política codiciosa derivada de $Q$ y la política de comportamiento es cualquier política *soft* derivada de $Q$.

**Algorithm 9** Q-learning (off-policy TD control) for estimating $\pi \approx \pi^*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$
Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
        $S \leftarrow S'$
    until $S$ is terminal

## 6.5. Diferencias entre Sarsa y Q-learning

Sarsa es un método on-policy, actualizando sus valores en función de las acciones de la política actual, que incluyen movimientos exploratorios. Q-learning, por otro lado, es un método off-policy, actualizando sus valores bajo la suposición de que se toma la mejor acción posible en cada paso.

En Sarsa, la estrategia de exploración influye directamente en el proceso de aprendizaje, lo que conduce a políticas más conservadoras que toman en cuenta los riesgos asociados con los movimientos exploratorios. Q-learning, al asumir acciones óptimas, aprende de manera más agresiva y puede manejar entornos de riesgo de manera más efectiva.

Sarsa es especialmente adecuado para escenarios donde la estrategia de exploración utilizada durante el aprendizaje tiene un impacto significativo en el rendimiento general. Esto es común en entornos altamente estocásticos, donde los resultados de las acciones son impredecibles, y en situaciones donde la seguridad o la gestión de riesgos son una prioridad. Q-learning, por otro lado, es ideal para escenarios donde el objetivo final es encontrar la mejor política posible, independientemente de las acciones que tome el agente durante el proceso de aprendizaje. Esto es común en juegos o tareas donde hay una alta variabilidad en las recompensas y donde el entorno es determinista o menos estocástico.

CAPÍTULO 7

# Métodos de Gradiente de Política

A diferencia de los métodos basados en el valor de acción, que estiman el valor de las acciones para informar indirectamente a la política, los métodos de gradiente de política (*Policy Gradient Methods*) aprenden un mapeo directo de estados a acciones a través de una política parametrizada $\pi(a|s, \theta)$. El objetivo central de estos métodos es optimizar una medida de rendimiento $J(\theta)$ con respecto a los parámetros de la política $\theta$, utilizando el ascenso por gradiente para mejorar iterativamente la política.

La forma general de la actualización del gradiente de política es:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t), \tag{7.1}$$

donde $\nabla J(\theta_t)$ es el gradiente de la medida de rendimiento, que estos métodos estiman y utilizan para ajustar los parámetros de la política.

## 7.1. Ventajas de los Métodos de Gradiente de Política

Los métodos de gradiente de política ofrecen varias ventajas, especialmente en entornos con espacios de acción grandes o continuos. A diferencia de los métodos basados en el valor de acción, los enfoques de gradiente de política permiten el aprendizaje de políticas estocásticas, lo cual puede ser crucial en entornos donde la estrategia óptima es inherentemente estocástica.

Además, estos métodos facilitan de forma natural el aprendizaje de políticas que se aproximan al determinismo de manera asintótica. En escenarios donde la política podría ser más sencilla de aproximar que la función de valor de acción, los métodos de gradiente de política suelen superar a los métodos tradicionales.

## 7.2. La Arquitectura Actor-Critic

Los métodos Actor-Critic son una subclase esencial de los métodos de gradiente de política que combinan los beneficios tanto del actor (política) como del crítico (función de valor). El *actor* en este contexto es la política aprendida, parametrizada por $\theta$, que selecciona acciones basadas en el estado actual. El "crítico" evalúa la acción tomada estimando la función de valor $V(s, w)$ con parámetros $w$.

El actor actualiza los parámetros de la política $\theta$ en una dirección sugerida por la evaluación del crítico, mientras que el crítico actualiza los parámetros de la función de valor $w$ para estimar mejor el retorno. Esta relación simbiótica permite que el actor mejore su política de manera informada, guiado por la evaluación del crítico del entorno.

Los métodos Actor-Critic, como una subclase de los enfoques de gradiente de política, ofrecen un marco poderoso para el aprendizaje por refuerzo. Al integrar el gradiente de política con la aproximación de la función de valor, proporcionan un enfoque equilibrado que combina las fortalezas de los métodos basados en el valor y los basados en la política.

CAPÍTULO 8

# Transfer Learning

El aprendizaje por transferencia (*Transfer Learning*) en el contexto del *Reinforcement Learning* es un enfoque que busca mejorar la eficiencia y efectividad del entrenamiento de un agente de RL al transferir conocimientos de una tarea previamente aprendida a una nueva tarea relacionada. Esto permite compartir el conocimiento adquirido entre los modelos, logrando una mejor eficiencia tanto en términos de recursos computacionales como de tiempo de entrenamiento. Al igual que con otras formas de aprendizaje automático, existe el riesgo del sobreajuste a la tarea de origen y no generalizar bien a la tarea de destino [2].

# Bibliografía

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. Ver página 3.

[2] UNIR, La Universidad en Internet. ¿qué es el transfer learning y qué ventajas tiene?, 2023. Ver página 37.