Aprendizaje por Refuerzo

Marta Caro Martínez

Aprendizaje por refuerzo

- Es una de las tres grandes **áreas del aprendizaje automático** junto al aprendizaje supervisado y no supervisado
- Funciona de distinta forma a los otros dos tipos de aprendizaje:
  - Modelo: **intento + error** al interactuar con el entorno
- El aprendizaje por refuerzo ha evolucionado rápidamente en los últimos años
  - Se aplica en **dominios muy variados**: aplicación en sistemas de recomendación, coches autónomos o videojuegos
  - **Deep reinforcement learning**: Deep learning + Aprendizaje por refuerzo

Elementos clave

- Agente
- Entorno
- Estado
- Acción
- Recompensa

Agente

- Es un software que aprende a tomar decisiones inteligentes
- Un agente es un **aprendiz** en el entorno del aprendizaje por refuerzo
- Ejemplos:
  - Jugador de ajedrez: aprende a realizar los mejores movimientos (decisiones) para ganar la partida
  - Mario en Super Mario Bros: Mario explora el juego y aprende a hacer las mejores jugadas

Entorno

- El entorno es el mundo del agente.
- El agente permanece dentro del entorno.
- Ejemplos:
  - Ajedrez: el entorno es el tablero, el ajedrecista (agente) aprende a jugar al ajedrez dentro del tablero (entorno).
  - Super Mario: el entorno es el mundo de Mario

Estado

El estado es la posición o movimiento en el que el agente está en el entorno en un momento determinado

- El agente siempre permanece dentro del entorno
- Puede haber muchas posiciones en el entorno en las que puede permanecer el agente, y esas posiciones se denominan **estados**.
- El estado se suele denotar como *s* (state)
- Ejemplo:
  - Ajedrez: en nuestra partida de ajedrez, cada posición del tablero se denomina estado.

Acción

- El agente interactúa con el entorno y pasa de un estado a otro realizando una **acción**.
- Normalmente se denota como *a* (action).
- Ejemplo:
  - Ajedrez: en el entorno del juego de ajedrez, la acción es el movimiento que realiza el jugador (agente).

Recompensa

- El agente interactúa con el entorno realizando una acción y pasa de un estado a otro.
- **En función de la acción, el agente recibe una recompensa.**
- Una recompensa no es más que un **valor numérico**, por ejemplo, +1 para una buena acción y -1para una mala acción.
- ¿Cómo decidimos si una acción es buena o mala?
- Ejemplo en el ajedrez:
  - si el agente hace un movimiento en el que elimina una de las piezas de ajedrez del oponente, entonces el agente recibe una **recompensa positiva (se considera una buena acción)**
  - si el agente hace un movimiento que lleva al contrincante a eliminar una de las piezas del agente, entonces el agente recibe una **recompensa negativa (se considera una mala acción)**

Idea básica del Aprendizaje por Refuerzo

- Ejemplo de analogía:
  - Le queremos enseñar a un perro a que coja una pelota
  - Le tiramos la pelota
  - Le damos una galleta cada vez que coge la pelota
  - El perro entenderá que recibe una galleta cada vez que coge la pelota
  - El perro aprenderá que cogiendo la pelota maximiza el número de galletas que consigue

Idea básica del Aprendizaje por Refuerzo

- En el aprendizaje por refuerzo:
  - **No enseñamos al agente qué tiene que hacer o cómo tiene que hacerlo**
  - Recibe una recompensa cada vez que realiza una acción
    - Buena acción → recompensa positiva
    - Mala acción → recompensa negativa
  - El agente empieza realizando una acción aleatoria
    - **Si la acción es buena** → recompensa positiva → el agente entiende que la acción es buena → **repetirá la acción**
    - **Si la acción es mala** → recompensa negativa → el agente entiende que la acción es mala → **no volverá a repetir la acción**

Idea básica del Aprendizaje por Refuerzo

- El aprendizaje por refuerzo es un **proceso de aprendizaje de prueba y error**:
  - El agente intenta diferentes acciones y aprende las buenas (que son las que le dan recompensas positivas)

Ejemplo:
queremos que un robot aprenda a caminar, pero sin ir a las montañas

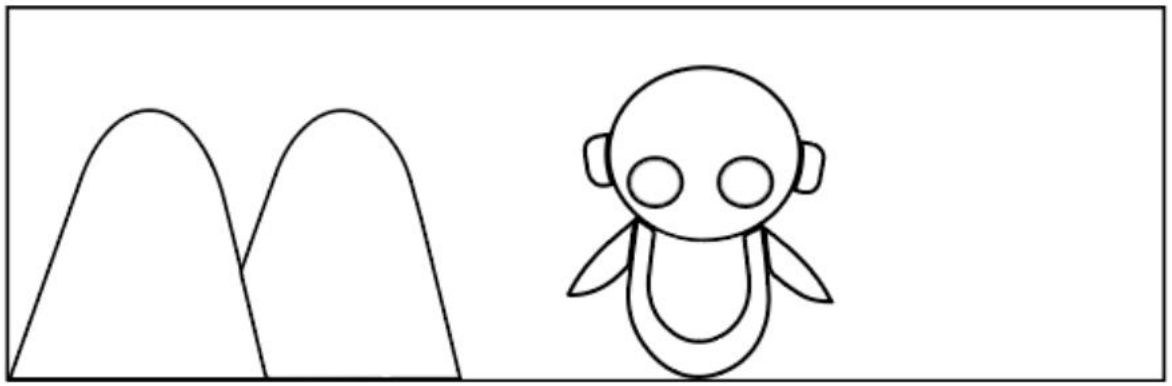


Figure 1.1: Robot walking

Idea básica del Aprendizaje por Refuerzo

- No le enseñamos al robot que no vaya en la dirección de las montañas
  - Si el robot se choca con las montañas, le damos una recompensa negativa (-1)
  - El robot entiende que es una acción incorrecta → no repetirá la acción

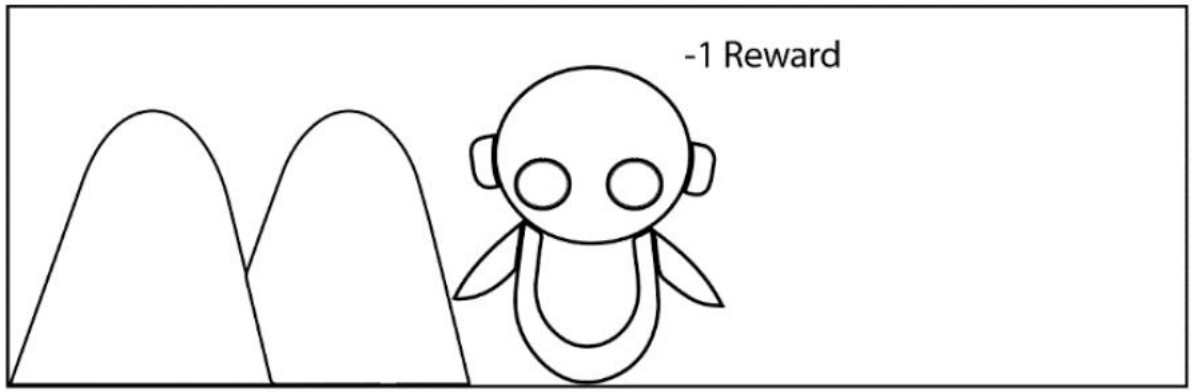


Figure 1.2: Robot hits mountain

Idea básica del Aprendizaje por Refuerzo

- No le enseñamos al robot que no vaya en la dirección de las montañas
  - Si el robot camina y no se choca con las montañas, le damos una recompensa positiva (+1)
  - El robot entiende que la respuesta es positiva, repetirá la acción

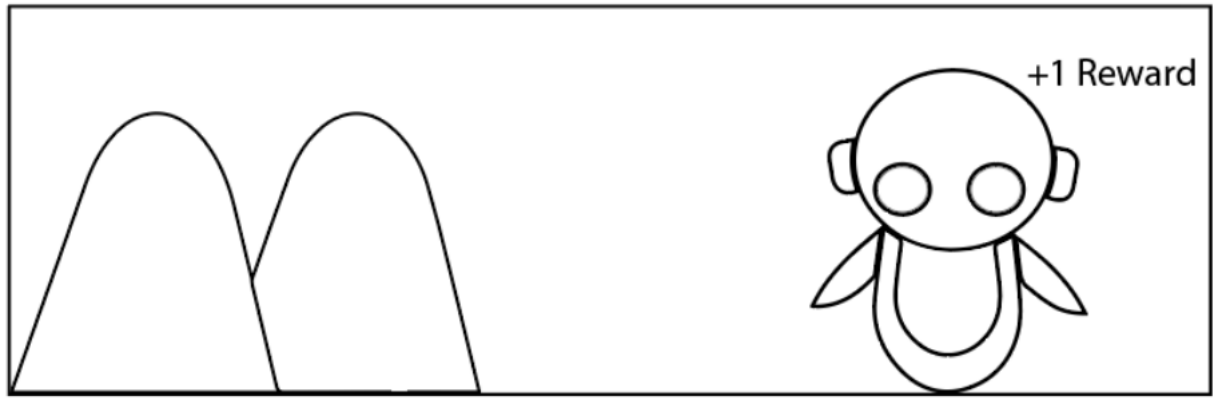


Figure 1.3: Robot avoids mountain

Algoritmo básico en Aprendizaje por Refuerzo

Objetivo principal del algoritmo: **maximizar la recompensa**

Los **pasos típicos** que se producen en algoritmos de aprendizaje por refuerzo son:

1. Primero, **el agente interactúa con el entorno** → realiza una acción
2. A través de esa acción, **el agente pasa de un estado a otro**
3. El **agente recibe una recompensa** dependiendo de la acción
4. Dependiendo de la recompensa, el agente entenderá si la acción es buena o mala
    1. Acción buena → **repetirá la acción**
    2. Acción mala → **cambiará de acción** buscando la recompensa positiva

Ejemplo del algoritmo

- La cuadrícula es el entorno
- Objetivo: el agente debe viajar de A a I sin pasar por las zonas con rayas (B, C, G, H)
- Si visita zonas con rayas: -1
- Si visita zonas sin rayas: +1
- Posibles acciones: arriba, abajo, izquierda y derecha

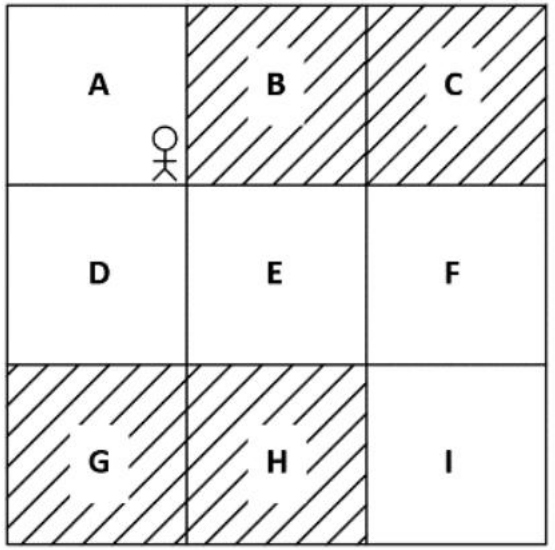


Figure 1.4: Grid world environment

Ejemplo del algoritmo

Primera iteración

- Movimientos aleatorios
- Moverse hacia la derecha desde A: recompensa negativa → mala acción
- Recompensas positivas → buenas acciones → moverse desde:
  - B a E
  - E a F
  - F a I

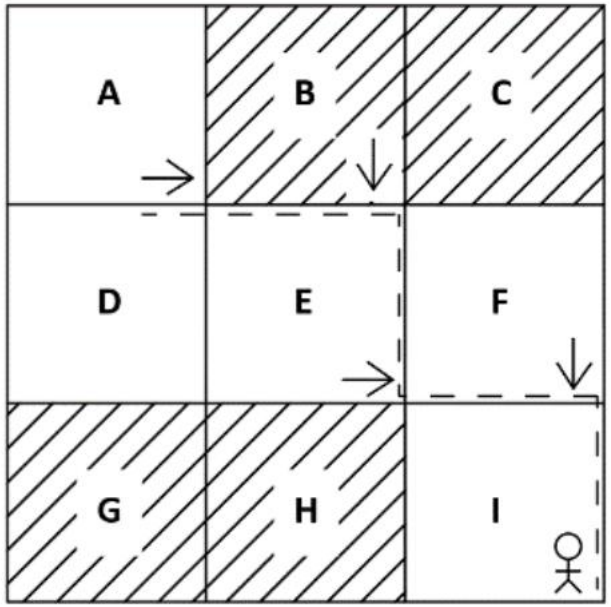


Figure 1.5: Actions taken by the agent in iteration 1

Ejemplo del algoritmo

Segunda iteración

- Desde A no se mueve a la derecha, prueba otra acción → va a D
- Movimientos positivos:
  - A a D
  - H a I
- Movimientos negativos:
  - D a G
  - G a H

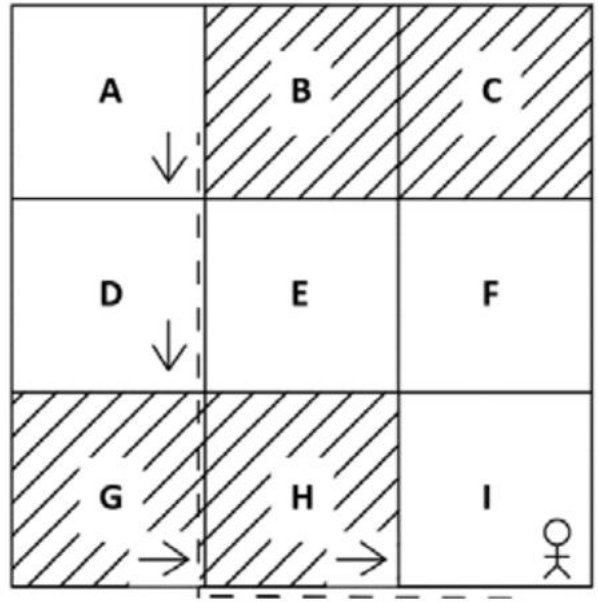


Figure 1.6: Actions taken by the agent in iteration 2

Ejemplo del algoritmo

Tercera iteración

- Gracias a las recompensas anteriores, el agente ha aprendido que:
  - Se debe de mover desde a A a D
  - No debe de moverse de D a G → se mueve de D a E
  - Se debe de mover de E a F
  - Se debe de mover de F a I

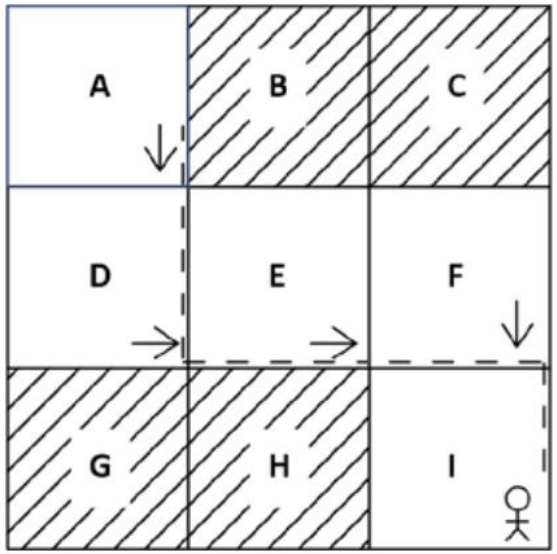


Figure 1.8: The agent reaches the goal state without visiting the shaded states

A estas iteraciones, se les llamada episodios

Diferencias con otros tipos de AA

RL vs AA supervisado

- **AA supervisado: datos + clases**
  - **Los datos guían al modelo**
  - El modelo puede generalizar su aprendizaje para predecir clases da datos que no ha visto anteriormente
- Ejemplo:
  - El perro y la pelota. En AA: le enseñamos al perro los movimientos que tiene que hacer. En RL: le damos galletas si coge la pelota
  - Ajedrez. En AA: le enseñamos los movimientos que los jugadores pueden hacer en cada estado. En RL: le damos un premio por cada acción positiva

Diferencias con otros tipos de AA

RL vs AA no supervisado

- AA no supervisado: datos
  - **El modelo encuentra patrones en los datos y su estructura**
  - **RL: se maximiza una recompensa**
- Ejemplo:
  - Sistema de recomendación → AA: encontramos películas que le han gustado a usuarios similares a mí. En RL: el usuario objetivo constantemente da feedback (recompensa) al sistema recomendador
    - Una recompensa puede ser un rating, minutos vistos de la película, etc.

Diferencias con otros tipos de AA

RL vs otros tipos de AA

- Otros tipos de AA: aprenden de los datos proporcionados
- **RL aprende como consecuencia de la interacción entre el agente y su entorno**

Markov Decision Processes

- El proceso de decisión de Markov (MDP) proporciona un marco matemático para resolver problemas de RL
- Casi todos los problemas de RL se pueden modelar como un MDP
- Veremos los conceptos de:
  - **Propiedad de Markov**
  - **Cadena de Markov**

La propiedad de Markov y la cadena de Markov

- La **propiedad de Markov** establece que el futuro solo depende del estado actual y no del pasado
- La **cadena de Markov** (o proceso de Markov) consiste en una secuencia de estados que obedecen a la propiedad de Markov
  - El proceso de Markov es un **modelo probabilístico que solo depende del estado actual para predecir el estado futuro**, sin tener en cuenta los estados pasados
- Ejemplo: si el tiempo es nublado, podemos predecir que el tiempo será lluvioso, sin tener en cuenta el tiempo anterior
  - IMPORTANTE: no funciona para todos los casos → tirar un dado: no depende del estado actual

El proceso de Markov

- **Transición**: cambiar de un estado al siguiente
- **Probabilidad de la transición** → P(s'|s): probabilidad de moverse de un estado s al siguiente estado s'
- **Tabla de Markov**: tabla donde se muestran todas las probabilidades de transición entre todos los estados
  - Ejemplo → estados: cloudy, rainy, windy

Current State	Next State	Transition Probability
Cloudy	Rainy	0.7
Cloudy	Windy	0.3
Rainy	Rainy	0.8
Rainy	Cloudy	0.2
Windy	Rainy	1.0

Table 1.1: An example of a Markov table

Tabla de Markov

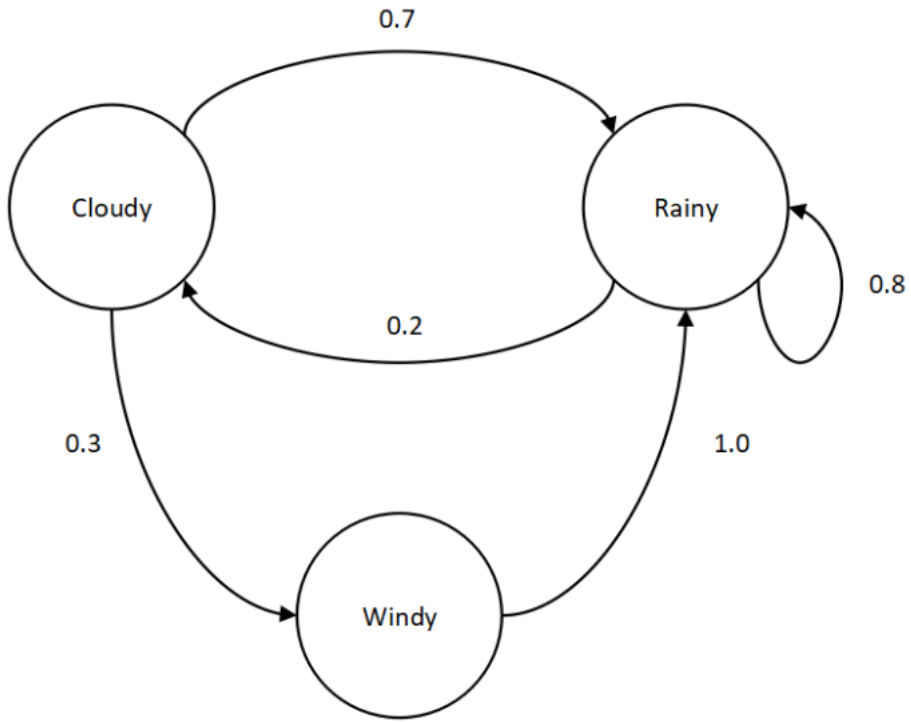


Figure 1.9: A state diagram of a Markov chain

Current State	Next State	Transition Probability
Cloudy	Rainy	0.7
Cloudy	Windy	0.3
Rainy	Rainy	0.8
Rainy	Cloudy	0.2
Windy	Rainy	1.0

Table 1.1: An example of a Markov table

$$
\begin{array}{c} \\ \text{Cloudy} \\ \text{Rainy} \\ \text{Windy} \end{array}
\begin{array}{c}
\begin{array}{ccc} \text{Cloudy} & \text{Rainy} & \text{Windy} \end{array} \\
\begin{bmatrix} 0.0 & 0.7 & 0.3 \\ 0.2 & 0.8 & 0.0 \\ 0.0 & 1.0 & 0.0 \end{bmatrix}
\end{array}
$$

Figure 1.10: A transition matrix

El proceso de Markov

- Comclusión: La cadena de Markov o el proceso de Markov consiste en un conjunto de estados junto con sus probabilidades de transición

El Proceso de Recompensa de Markov (MRP)

- El proceso de recomensa de Markov (MRP) es una extensión del proceso de Markov, pero incluye una función de recompensa
- El proceso de recompensa de Markov incluye:
  - Estados (s)
  - Probabilidad de transición de cada estado (s'|s)
  - **Función de recompensa de cada estado**
    - **La recompensa obtenida en cada estado**
    - Normalmente se denota como R(s)

El Proceso de Decisión de Markov (MDP)

- Es una extensión del MRP, pero incluyendo acciones
- El proceso de decisión de Markov incluye:
  - Estados (s)
  - Probabilidad de transición de cada estado (s'|s)
  - Función de recompensa de cada estado (R(s))
  - **Acciones**
- **La propiedad de Markov se puede aplicar sobre un problema de aprendizaje por refuerzo**
  - El agente solo toma decisiones basadas en el estado actual y no basadas en los estados pasados

Proceso de Decisión de Markov (MDP) - Ejemplo

- Modelamos el ejemplo anterior como un MDP
  - **Estados**: A – I
  - **Acciones**: arriba, abajo, izquierda, derecha
  - **Probabilidad de transición** *P(s'|s,a)* de un estado *s* al siguiente *s'* cuando ejecutamos la acción *a*. Incluimos las acciones en el cálculo de las probabilidades (en MRP no las usábamos).

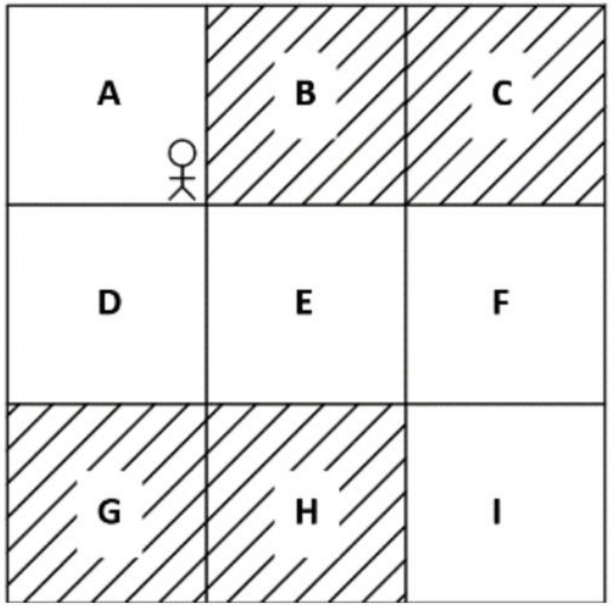
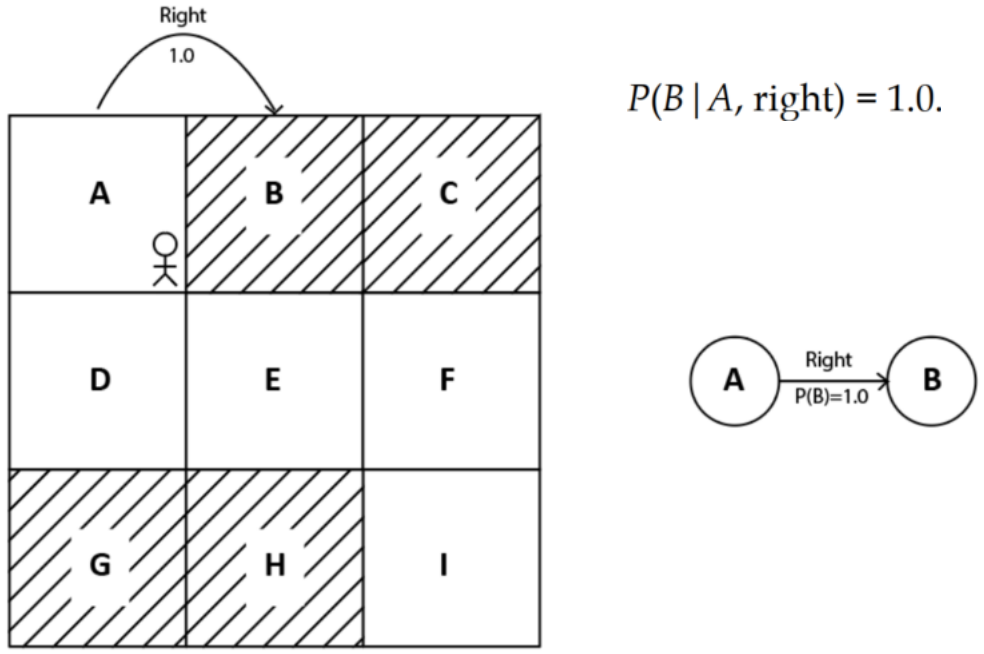


Figure 1.11: Grid world environment

Proceso de Decisión de Markov (MDP) - Ejemplo

- Ejemplos de probabilidad de transiciones



$P(B \mid A, \text{right}) = 1.0.$

Figure 1.12: Transition probability of moving right from A to B

$P(F \mid C, \text{down}) = 0.9.$

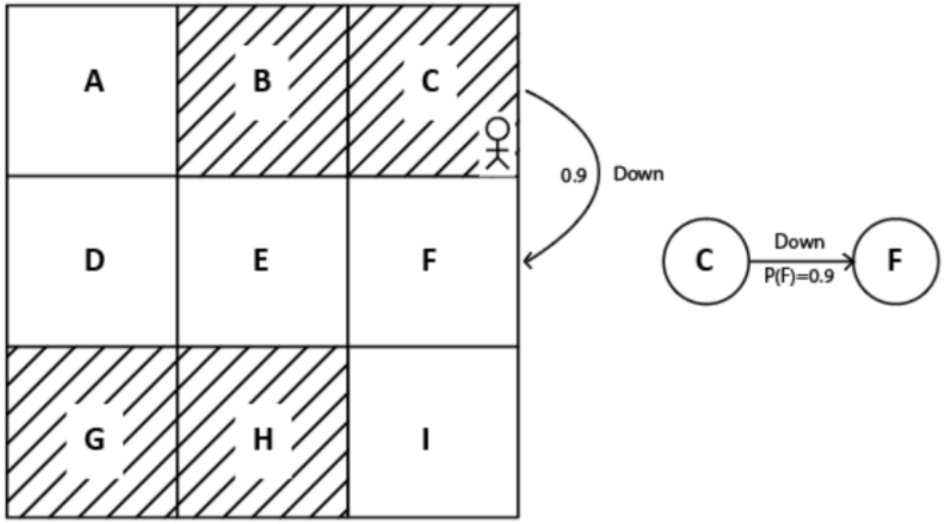


Figure 1.13: Transition probability of moving down from C to F

Proceso de Decisión de Markov (MDP) - Ejemplo

- Modelamos el ejemplo anterior como un MDP
  - **Estados**: A – I
  - **Acciones**: arriba, abajo, izquierda, derecha
  - **Probabilidad de transición** *P(s'|s,a)* de un estado *s* al siguiente *s'* cuando ejecutamos la acción *a*. Incluimos las acciones en el cálculo de las probabilidades (en MRP no las usábamos).
  - **Función de recompensa** *R(s,a,s')*: representa la recompensa que obtiene un agente cuando se mueve del estado *s* al estado *s'* al ejecutar la acción *a*.

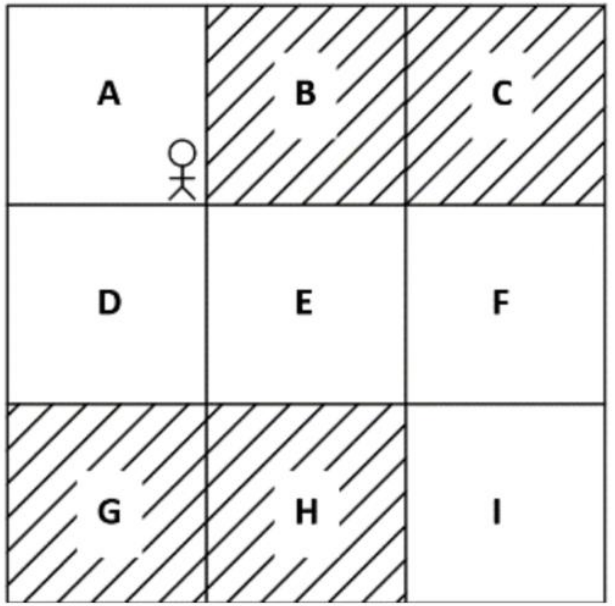


Figure 1.11: Grid world environment

Proceso de Decisión de Markov (MDP) - Ejemplo

- Ejemplos de recompensas

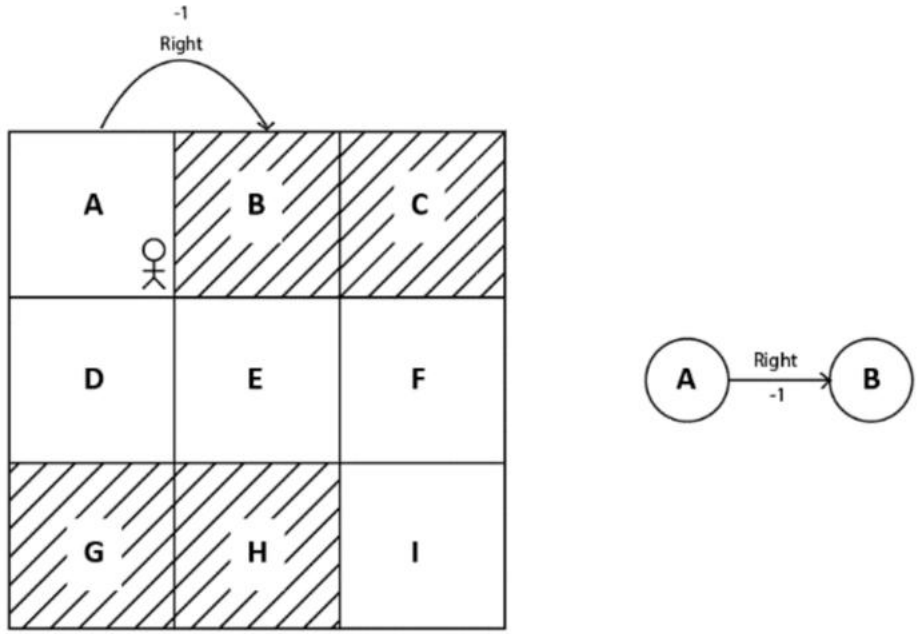


Figure 1.14: Reward of moving right from A to B

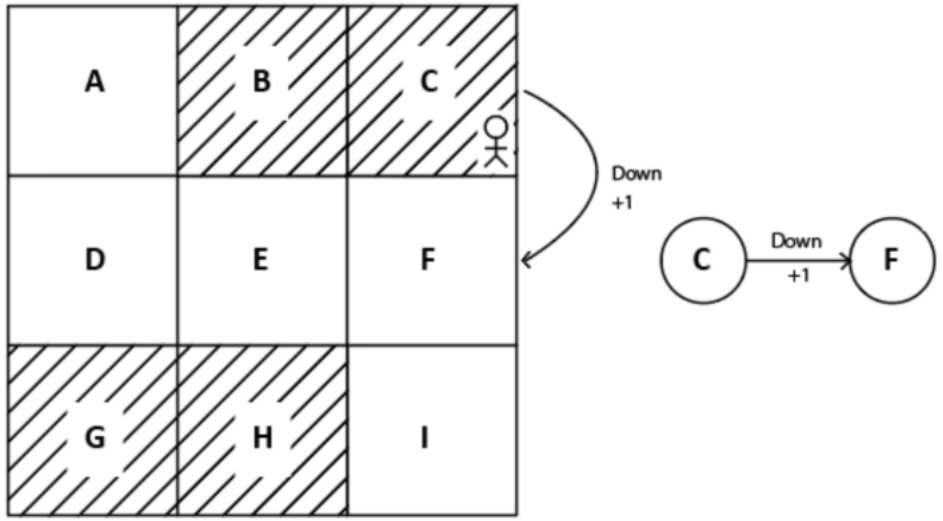


Figure 1.15: Reward of moving down from C to F

Proceso de Decisión de Markov (MDP)

- Conclusióm: El proceso de decisión de Markov se usa para modelar problemas de aprendizaje por refuerzo

Conceptos básicos

- Expectativa
- Espacio de acción
- Política
- Episodio
- Tipos de tareas
- Horizonte
- Retorno y factor de descuento
- Función de valor
- Función Q
- Tipos de aprendizaje
- Tipos de entornos

Expectativa

- La expectativa es el **valor esperado** de un valor futuro:
  - **Promedio ponderado de posibles resultados basado en probabilidades**
  - Se aplica en RL para calcular la recompensa que un agente puede recibir considerando las distintas acciones que se pueden producir
- A la derecha → Ejemplos de cálculo de expectativas de una variable aleatoria X

Al lanzar un dado

X	1	2	3	4	5	6
P(x)	1/6	1/6	1/6	1/6	1/6	1/6

Table 1.2: Probabilities of throwing a dice

$$E(X) = \sum_{i=1}^{N} x_i p(x_i)$$

$$E(X) = 1(1/6) + 2(1/6) + 3(1/6) + 4(1/6) + 5\,(1/6) + 6(1/6) = 3.5.$$

Al aplicar una función al lanzamiento del dado

X	1	2	3	4	5	6
f(x)	1	4	9	16	25	36
P(x)	1/6	1/6	1/6	1/6	1/6	1/6

Table 1.3: Probabilities of throwing a dice

$$\mathbb{E}_{x\sim p(x)}[f(X)] = \sum_{i=1}^{N} f(x_i)p(x_i)$$

$$E(f(X)) = 1(1/6) + 4(1/6) + 9(1/6) + 16(1/6) + 25(1/6) + 36(1/6) = 15.1.$$

Espacio de acción

- **Es el conjunto de las posibles acciones a realizar en el entorno**
  - Espacio de acción en el ejemplo de la cuadrícula anterior: [arriba, abajo, izquierda, derecha]
- Dos tipos de espacios de acción:
  - **Discreto**: cuando las acciones del espacio son contables y finitas. El ejemplo anterior es discreto
  - **Continuo**: el conjunto de acciones que un agente puede realizar es infinito y continuo.
    - Ejemplo: cuando queremos entrenar un agente para conducir un coche, el espacio consistirá en acciones con valores continuos (velocidad, ángulo de rotación del volante…)

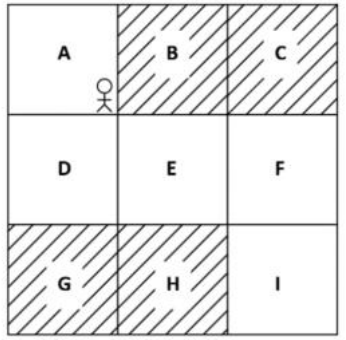


Figure 1.16: Grid world environment

Política

- **Una política define el comportamiento del agente en el entorno**
  - La política le dice al agente qué acciones realizar en cada estado
- La primera vez, se inicializa una política aleatoria
- En las siguientes iteraciones, el agente aprenderá las mejores acciones (las que dan lugar a recompensas positivas)
  - **Esta política es la política óptima**
- Hay dos tipos de políticas:
  - Determinista
  - Estocástica

Optimal Policy

State	Action
A	Down
D	Right
E	Right
F	Down

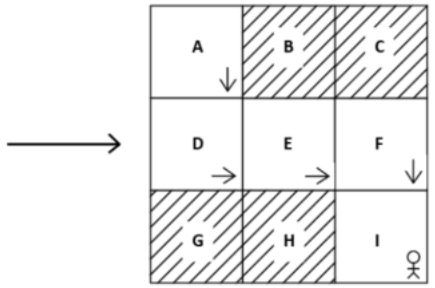


Figure 1.17: The optimal policy in the grid world environment

Política determinista

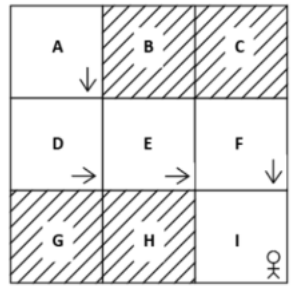
- **La política determinista le dice al agente cual es la acción (<u>solo una</u>) que debe realizar en un estado concreto**
  - Dado un estado *s* en un momento *t*, la política le dice al agente que tiene que realizar la acción *a*:

$$a_t = \mu(s_t)$$

  - En el ejemplo, cada vez que el agente esté en la celda A, realizará la acción abajo:

Optimal Policy

State	Action
A	Down
D	Right
E	Right
F	Down

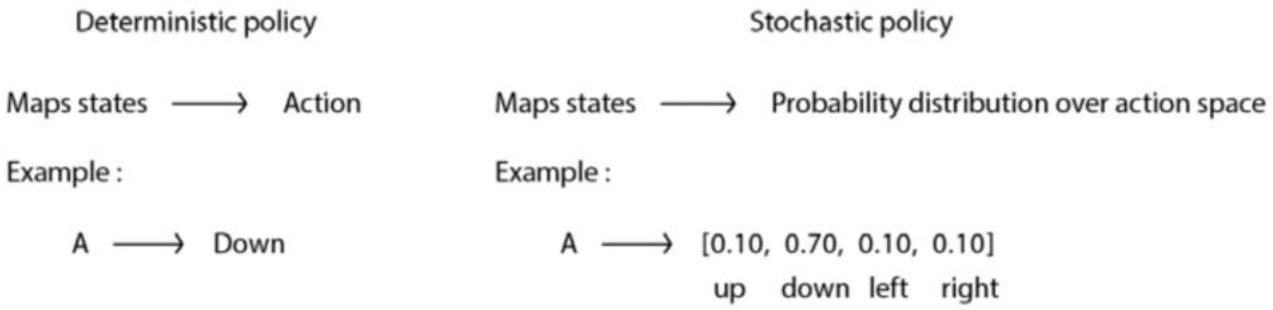


$$\mu(A) = \text{Down}$$

Figure 1.17: The optimal policy in the grid world environment

Política estocástica

- **La política estocástica** no asigna un estado directamente a una acción en particular; en su lugar, **asigna el estado a una distribución de probabilidad sobre un espacio de acciones**
  - **En la determinista**, siempre que el agente se encuentra en un estado, irá al siguiente estado (**siempre al mismo**) determinado por la política
  - **En la estocástica**, cada vez que el agente se encuentre en un estado **no se moverá al mismo**



Esto significa que cada vez que el agente visite el estado A, en lugar de moverse siempre al mismo estado, el 10% de las veces, el agente irá arriba, el 70%, abajo, etc.

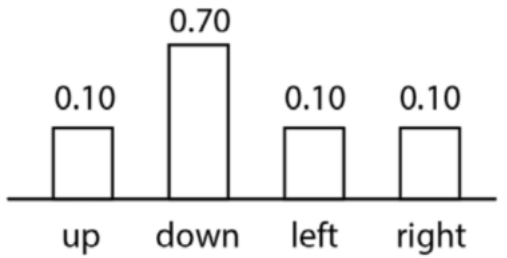
Política estocástica

- Se denota como $a_t \sim \pi(s_t)$ o $\pi(a_t|s_t)$
- Hay dos tipos:
  - **Política categórica**:

    El **espacio de acción es discreto**, las probabilidades son fijas para cada acción
  - **Política gaussiana**:

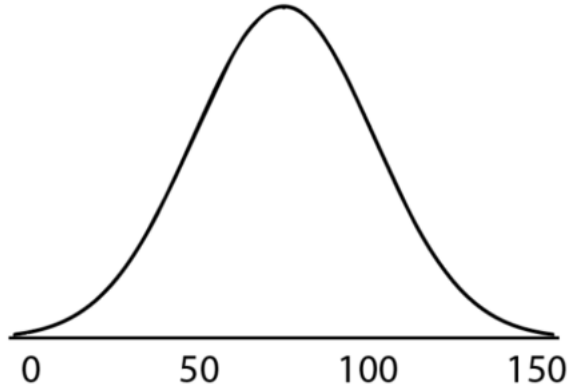
    El **espacio de acción es continuo**, las probabilidades son una distribución gaussiana para cada acción

Política categórica



Probabilidad del siguiente movimiento desde el estado A en un espacio de acción discreto

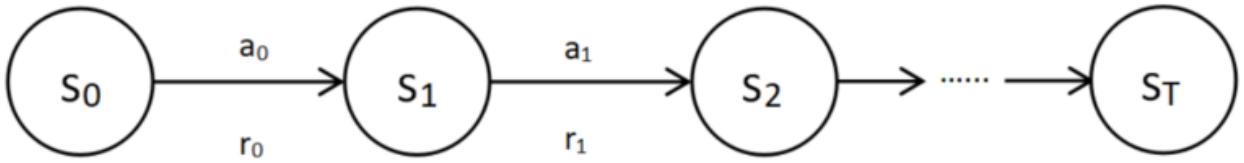
Política gaussiana



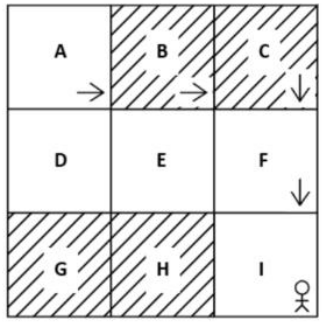
Probabilidad que los valores de velocidad (al conducir un coche de forma autónoma) tomarían

Episodio

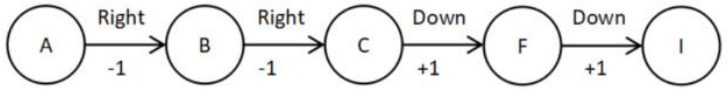
- Es la **trayectoria (conjunto de acciones)** que el agente toma desde el estado inicial hasta el estado final
- El agente va a llevar a cabo varios episodios
  - **Objetivo: encontrar la política óptima**
- El episodio necesita información de cada estado, cada acción y cada recompensa



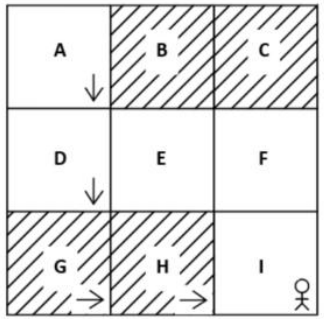
Episodio - Ejemplo



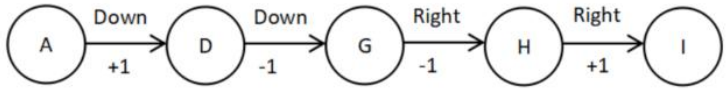
Episodio 1

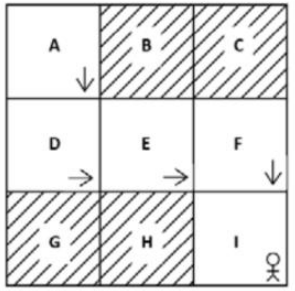
$\tau_1 =$ 

Episodios intermedios...

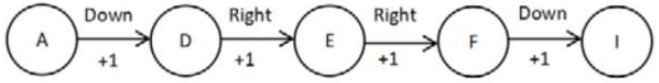


Episodio 2

$\tau_2 =$ 



Episodio N

$\tau_n =$ 

Tareas episódicas y continuas

- Una tarea de aprendizaje por refuerzo se puede categorizar como:
  - **Tarea episódica**: tienen episodios, y cada episodio tiene un estado final. Por ejemplo, una carrera en Mario Kart (el final es ganar o perder)
  - **Tarea continua**: no tienen episodios y no tienen estados finales. Por ejemplo, un robot asistente personal no tiene un estado final, continúa trabajando hasta que se apaga

Horizonte

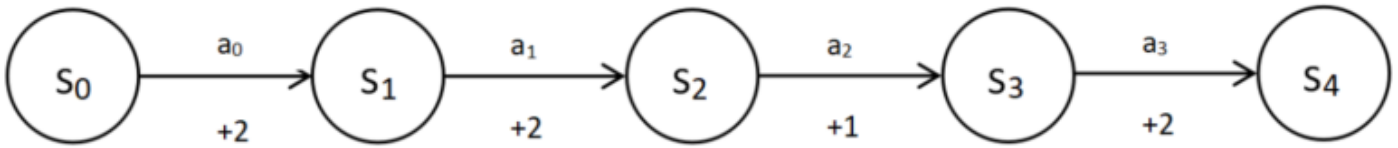
- El horizonte es **el momento en el que el agente termina** su interacción con el entorno
- Hay dos tipos:
  - **Horizonte finito**: el agente termina de interactuar con el entorno en un momento concreto → cuando es una tarea episódica
  - **Horizonte infinito**: el agente nunca termina de interactuar con el entorno → cuando es una tarea continua

Return (retorno)

- Es la suma de las recompensas obtenidas por un agente en un episodio. Se denota con *R* o *G*.

$$R(\tau) = r_0 + r_1 + r_2 + \ldots + r_T$$

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



R = 2 + 2 + 1 + 2 = 7

Factor de descuento

- **Objetivo de nuestro agente → maximizar el retorno**
- Maximizamos el retorno si el agente elige las acciones correctas en cada estado
- Para ello, usamos la política óptima
- **Conclusión → con la política óptima maximizamos el retorno**
- En tareas continuas: $R(\tau) = r_0 + r_1 + r_2 + \ldots + r_\infty$ → estamos intentado maximizar una suma que tiende a infinito
  - Solución: **factor de descuento (γ)**

$$R(\tau) = \gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_\infty \qquad R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

Factor de descuento

$$R(\tau) = \gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_\infty \qquad R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

El valor del factor de descuento está en [0,1]

Sin embargo, los valores óptimos están en [0.2,0.8]

Si $\gamma = 0$, el retorno es la primera recompensa, no tenemos en cuenta los pasos futuros

$$\begin{aligned} R &= (\gamma)^0 r_0 + (\gamma)^1 r_1 + (\gamma)^2 r_2 + \ldots \\ &= (0)^0 r_0 + (0)^1 r_1 + (0)^2 r_2 + \ldots \\ &= (1) r_0 + (0) r_1 + (0) r_2 + \ldots \\ &= r_0 \end{aligned}$$

Si $\gamma = 1$, el retorno tiende a infinito

$$\begin{aligned} R &= (\gamma)^0 r_0 + (\gamma)^1 r_1 + (\gamma)^2 r_2 + \ldots \\ &= (1)^0 r_0 + (1)^1 r_1 + (1)^2 r_2 + \ldots \\ &= r_0 + r_1 + r_2 + \ldots \end{aligned}$$

Factor de descuento

$$R(\tau) = \gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_\infty \qquad R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

Si el **tamaño del factor de descuento es pequeño** (alrededor de 0,2), irá decrementándose significativamente en cada paso futuro:

- Estamos **dando más importancia a la recompensa futura inmediata**

$$\begin{aligned} R &= (\gamma)^0 r_0 + (\gamma)^1 r_1 + (\gamma)^2 r_2 + \ldots \\ &= (0.2)^0 r_0 + (0.2)^1 r_1 + (0.2)^2 r_2 + \ldots \\ &= (1) r_0 + (0.2) r_1 + (0.04) r_2 + \ldots \end{aligned}$$

Factor de descuento

$$R(\tau) = \gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_\infty \qquad R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

Si el **tamaño del factor de descuento es grande** (alrededor de 0,8), se decrementa tan rápido en cada paso futuro:

- Estamos **dando más importancia a las recompensas futuras más lejanas**

$$\begin{aligned} R &= (\gamma)^0 r_0 + (\gamma)^1 r_1 + (\gamma)^2 r_2 + \ldots \\ &= (0.9)^0 r_0 + (0.9)^1 r_1 + (0.9)^2 r_2 + \ldots \\ &= (1) r_0 + (0.9) r_1 + (0.81) r_2 + \ldots \end{aligned}$$

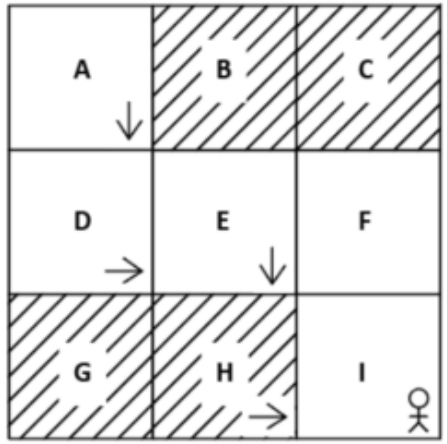
Factor de descuento

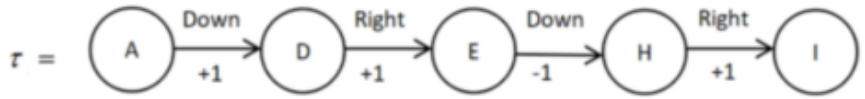
$$R(\tau) = \gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_\infty \qquad R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

- Dependiendo de la tarea de aprendizaje por refuerzo, **en unos casos nos beneficia más establecer un factor de descuento más pequeño o más grande**
  - En el ajedrez, el objetivo es matar al rey: necesitamos tener en cuenta las recompensas futuras
  - Si implementamos un sistema de frenado automático, nos interesa más dar prioridad a la recompensa inmediata (frenar), en lugar de las recompensas futuras (llegar más rápido al destino, ahorrar gasolina…)

Función valor

- Es la función que nos va a decir el **retorno** que un agente va a obtener **desde el estado s hasta el estado final** $V^{\pi}(s) = [R(\tau)|s_0 = s]$





$V(A) = 1+1+ -1+1 = 2$

$V(D) = 1-1+1= 1$

$V(E) = -1+1 = 0$

$V(H) = 1$

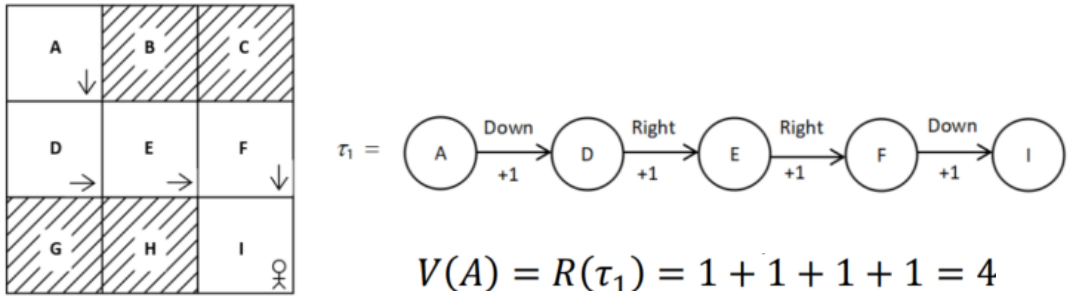
Función valor

- Si la política es estocástica, tendremos un conjunto de probabilidades de acción en cada estado, por lo que se necesita adaptar la función de valor
- Podemos usar el concepto de **expectativa**
- **La función valor nos devolverá el retorno esperado teniendo en cuenta las probabilidades de cada acción**

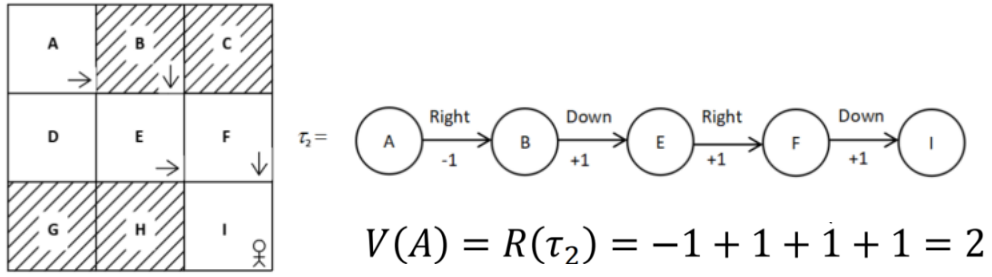
$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

Función valor - Ejemplo

- Estamos en el estado A y queremos calcular la función valor
- Tenemos la distribución de probabilidad sobre el espacio de acción: [0.0,0.80,0.00,0.20]
  - El 80% de las veces → acción abajo (imagen de la izquierda)
  - El 20% de las veces → acción derecha (imagen de la derecha)



$$V(A) = R(\tau_1) = 1 + 1 + 1 + 1 = 4$$



$$V(A) = R(\tau_2) = -1 + 1 + 1 + 1 = 2$$

Función valor - Ejemplo

- Acción Abajo: V(A) = 4 (80% de las veces)
- Acción Derecha: V(A) = 2 (20% de las veces)
- Aplicando la función valor:

$$V^{\pi}(A) = \mathop{\mathbb{E}}_{\tau \sim \pi}[R(\tau)|s_0 = A]$$

$$= \sum_i R(\tau_i)\pi(a_i|A)$$

$$= R(\tau_1)\pi(\text{down}|A) + R(\tau_2)\pi(\text{right}|A)$$
$$= 4(0.8) + 2(0.2)$$
$$= 3.6$$

Por tanto, **el valor de un estado es el retorno esperado de la trayectoria empezando desde ese estado**

Función valor

- La función valor depende de la política que usemos
- Tendremos diferentes valores según la política
- Se elige la política que maximice la función valor **Política óptima** $V^*(s) = \max_{\pi} V^{\pi}(s)$
- Ejemplo: tenemos dos políticas, con las que calculamos la función valor para el estado *s*. Elegiremos la política 1 en el ejemplo:

$$V^{\pi_1}(s) = 13 \qquad V^{\pi_2}(s) = 11$$

- También podemos mostrar los valores en una tabla. En el ejemplo, si tenemos dos estados, s0 y s1, conviene más quedarse en el estado s1 (**estado óptimo**)

State	Value
s_0	7
s_1	11

Función Q (función de valor estado-acción)

- Es la función que determina el valor de un estado-acción:
  - Es **el retorno que el agente obtiene desde el estado inicial *s*, realizando la acción *a* y siguiendo la política π**

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

- La diferencia con la función valor es que la función Q tiene en cuenta la acción *a* tomar en *s*

Función Q - Ejemplo

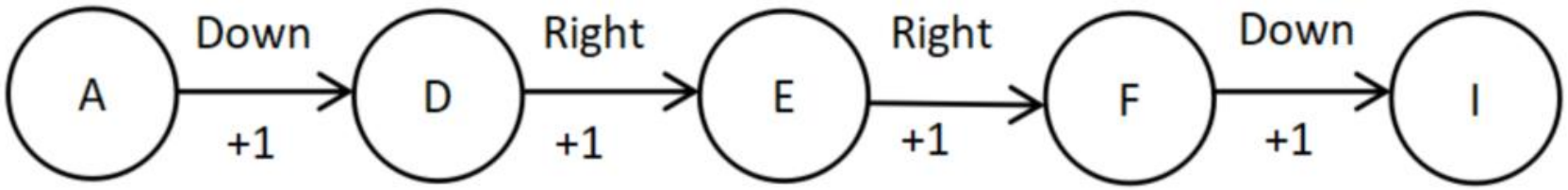


Figure 1.29: A trajectory/episode example

- Queremos calcular la función Q para A-abajo:

$$Q^{\pi}(A, \text{down}) = [R(\tau)|s_0 = A, a_0 = \text{down}]$$

$$Q(A, \text{down}) = 1 + 1 + 1 + 1 = 4$$

- La función Q para D-derecha:

$$Q^{\pi}(A, \text{right}) = [R(\tau)|s_0 = D, a_0 = \text{right}]$$

$$Q(A, \text{right}) = 1 + 1 + 1 = 3$$

Función Q

- Al igual que con la función valor, el valor de la función Q no toma valores fijos, puede tomar distintos valores de retorno
  - De nuevo usamos el concepto de expectativa: la función Q es el valor de **retorno esperado** que el agente obtendría desde el estado s ejecutando la acción a usando la política π

$$Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]$$

  - La función Q depende de la política: elegiremos la política que **maximice** el valor de la función Q

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

Función Q - Ejemplo

- También se pueden representar los valores de la función Q en una tabla:

State	Action	Value
s_0	0	9
s_0	1	11
s_1	0	17
s_1	1	13

Table 1.5: Q table

- En el ejemplo tenemos dos estados s0 y s1, y dos posibles acciones 0 y 1.
- Según los datos del ejemplo, la política óptima sería:
  - Realizar la acción 1 en el estado 0
  - Y realizar la acción 0 en el estado 1

Conclusión: podemos extraer la política óptima calculando los valores de la función Q

Tipos de aprendizaje

- Aprendizaje **basado en modelos**:
  - El agente tiene una descripción completa del entorno, conoce su dinámica:
    - La probabilidad de moverse de un estado *s* a un estado *s'* usando la acción *a*
    - La función recompensa obtendrá el valor de la recompensa al realizar ese movimiento
    - **El agente usa las dinámicas del modelo para obtener la política óptima**
- Aprendizaje **libre de modelos**:
  - El agente no conoce las dinámicas del modelo
    - **Obtiene la política óptima sin la descripción del entorno**

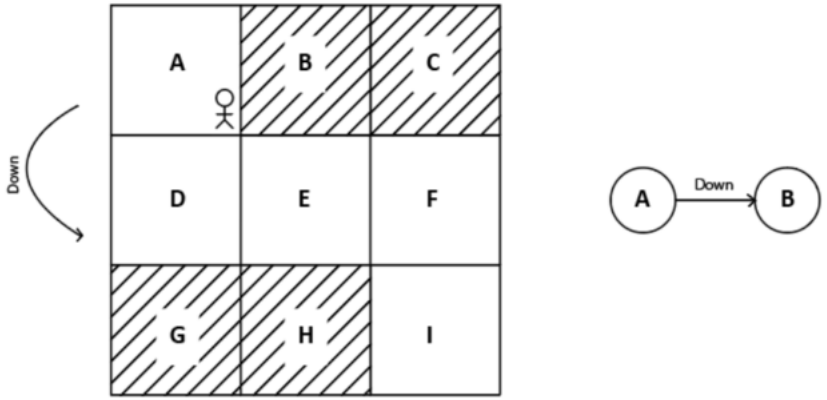
Tipos de entorno

El entorno es el mundo del agente:

- Determinista / Estocástico
- Discreto / Continuo
- Episódico / No episódico
- De un solo agente / Multiagente

Entorno determinista

- En un entorno determinista, estamos seguros de que cuando un agente realiza una acción *a* en un estado *s*, siempre llegara al estado s’
- Ejemplo: en este entorno, si el agente está en el estado A y se mueve hacia abajo, siempre llegará al estado D



Entorno estocástico

- En un entorno estocástico, no podemos decir que cuando un agente realiza una acción *a* en un estado *s*, siempre llegara al estado s'
  - Hay **aleatoriedad** asociada al entorno estocástico

Ejemplo: en este entorno, si el agente se encuentra en el estado A:

- 30% de las veces irá a B
- 70% de las veces irá a D

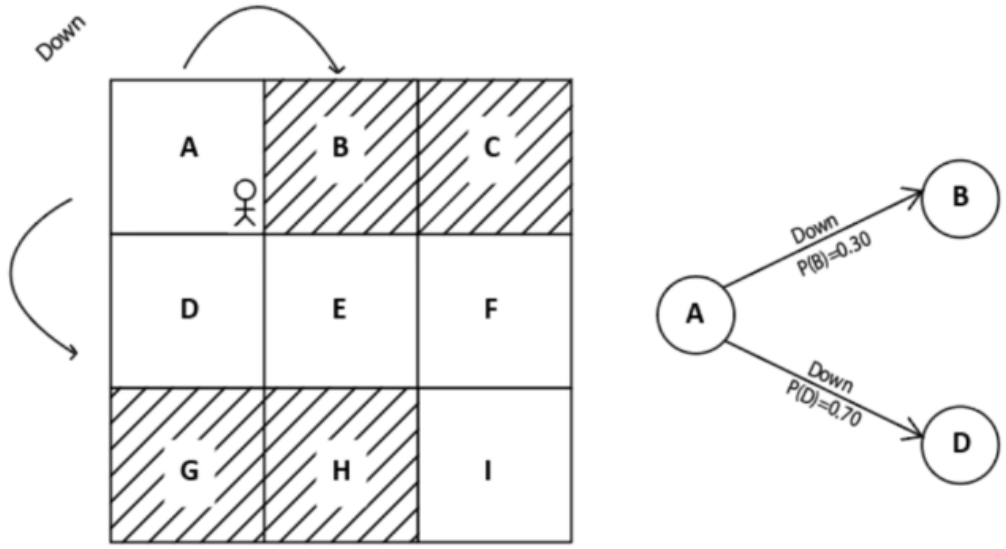


Figure 1.31: Stochastic environment

Entorno discreto / continuo

- **Entorno discreto**: el espacio de acción es discreto (fijo). Ejemplo: las acciones que se pueden realizar son [arriba, abajo, izquierda, derecha]
- **Entorno continuo**: el espacio de acción es continuo (infinitas posibilidades). Ejemplo al conducir un coche (velocidad del coche, ángulo del volante…)

Entorno episódico / no episódico

- **Entorno episódico (entorno no secuencial)**: la acción actual del agente no afectará a las acciones futuras
  - En clasificación de imágenes, no afectará la clasificación de una imagen a la clasificación de la siguiente
- **Entorno no episódico (secuencial)**: las acciones actuales del agente afectarán a las acciones futuras
  - El tablero del ajedrez en un entorno secuencial: las acciones del jugador afectarás a sus acciones futuras

Entorno de un solo agente / multiagente

- **Entorno de un solo agente**: solo hay un agente
- **Entorno multiagente:** varios agentes

Q learning

- Hay muchos algoritmos en aprendizaje por refuerzo que se basan en los conceptos que hemos visto anteriormente → uno de los más usados es el algoritmo de **Q learning**
- Principales características de Q learning:
  - **Aprende la función Q(s,a) usando TD Learning** (Aprendizaje por Diferencia Temporal): actualiza los valores de Q de forma independiente en cada paso, sin esperar a que termine el episodio

Q learning

- Principales características de Q learning:
  - Es un **método off-policy** → se usan dos políticas:
    - Una política para comportarse en el entorno (**exploración** del entorno)
      - Se usa en las primeras fases cuando el agente no sabe tomar las mejores decisiones
      - Ej: un robot en un laberinto probando una nueva ruta a ver si hay una salida más rápida
    - Otra política a optimizar para obtener la mejor recompensa (**explotación**):
      - Se usa en las últimas fases, de forma que siga las mejoras decisiones aprendidas
      - Ej: un agente en un tablero de ajedrez usa un movimiento que ya ha utilizado antes

Algoritmo de Q learning

1. Inicializa la función *Q(s, a)* con valores aleatorios
2. Para cada paso en el episodio:
   1. Extrae una **política** de *Q(s,a)* y selecciona una acción *a* a ejecutar en el estado *s*
   2. Se ejecuta la acción *a*, el agente se mueve al estado *s'*, y observa la recompensa *r'*
   3. Se actualiza la función *Q*:

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s'a') - Q(s,a)\right)$$

   4. Se actualiza el estado al siguiente *s* = *s'*
   5. Si *s'* no es un estado final del episodio, repetir los pasos 1 al 5

Algoritmo de Q learning

- En el algoritmo de Q-learning, la **política** del paso 2.1 va cambiando:
  - En las primeras fases se usa una **política epsilon-greedy**:
    - Con probabilidad épsilon, el agente elige una acción aleatoria (exploración).
    - Con probabilidad 1 – épsilon, el agente elige la mejor acción conocida (explotación → usando la **política greedy**).
      - La política greedy consiste en elegir la acción que maximiza el valor de Q
  - Conforme avanza el aprendizaje, se reduce épsilon gradualmente para favorecer el aprendizaje
  - En las últimas fases, se usa una política más greedy para aprovechar el conocimiento aprendido

Algoritmo de Q learning

(Ecuación de Bellman)

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s'a') - Q(s,a)\right)$$

- Actualización de Q(s,a):
  - $Q(s,a)$ es el valor de ejecutar *a* en el estado *s* antes de actualizar
  - $\left(r + \gamma \max_{a'} Q(s'a') - Q(s,a)\right)$ es la nueva estimación de la recompensa futura: *r* es la recompensa inmediata, y $\gamma \max_{a'} Q(s'a') - Q(s,a)$ es el valor descontado de las futuras recompensas, teniendo en cuenta el siguiente estado *s'* y la mejor acción posible *a'* en ese estado (γ es el factor de descuento)
  - $Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s'a') - Q(s,a)\right)$ es la actualización del valor de Q(s,a): sumamos la diferencia entre la nueva estimación de la recompensa y el valor actual, multiplicándola por una tasa de aprendizaje, con el valor actual

Deep Learning + Q-learning

- Cuando los espacios son arbitrariamente grandes, las tablas de valores Q son inmensas e impracticables (ej: videjouegos o robótica)
  - Solución: usar redes neuronales para calcular los valores de Q(s,a) de cada acción
- Se aplica a Q-learning: **Deep Q-learning**
  - La red va a recibir una imagen (de un juego, por ejemplo) o un conjunto de variables (entorno, estado, acciones, etc.)
  - Va a devolver un valor *Q(s,a)* para cada acción posible
    - El entrenamiento se produce usando información de un conjunto aleatorio de experiencias pasadas guardadas en una **memoria de repetición**
    - El objetivo de la red es acercar Q(s,a) al valor que se obtendría con la ecuación de Bellman

Librería para implementación de RL

: https://gymnasium.farama.org/index.html