Aprendizaje por Refuerzos

Diplomatura en Ciencia de Datos, Aprendizaje Automático y sus Aplicaciones - FaMAF, 2022

Sobre nosotros

Jorge Palombarini

- Data Science Technical Leader @ Mercadolibre

Juan Cruz Barsce

- Data Scientist @ Mercadolibre

Agenda

- Introducción
- Nociones básicas
- Ejemplos de políticas y MDPs
- Formas de resolverlos
- Bandidos
- Diferencia temporal
- Métodos basados en modelos
- Ética

Introducción

Introducción

¿Aprendizaje por refuerzos? (*reinforcement learning,* ***RL***)

Una forma propuesta: es una **sub-área de *machine learning*** (*ML*) y combina elementos de las demás

Aprendizajes en ML (muy simplificadamente)

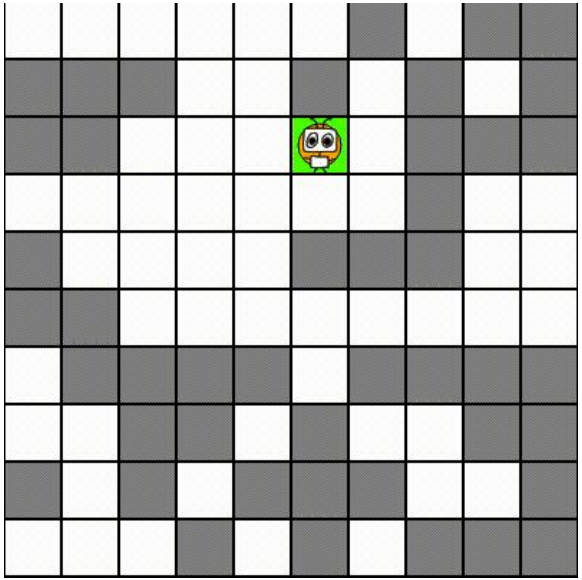
- Supervisado: encontrar $f : X \to y$
- No supervisado: encontrar $f : X \to X'$
- Por refuerzos: encontrar $f : \mathrm{S} \to A$ (óptimas) a partir de X que es generado con experiencia

Introducción

¿Aprendizaje por refuerzos?

Otra forma propuesta, como un **área por encima de ML**, que es necesaria como forma de generalizar la IA (ej. AIXI de Marcos Hutter)

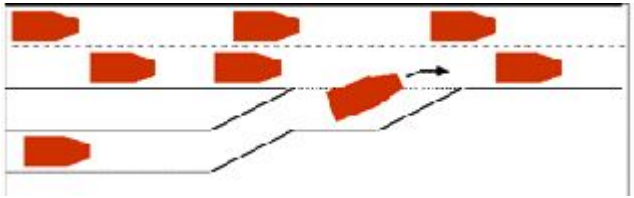
Demo de AIXI

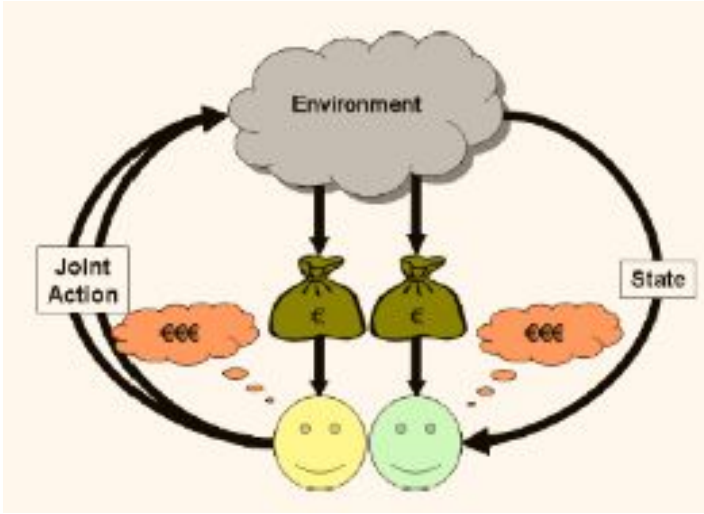


Introducción

¿Aprendizaje por refuerzos?

Otra forma propuesta: como un **área aparte que estudia el comportamiento de agentes**, siendo RL la forma más simple en la que se considera un único agente

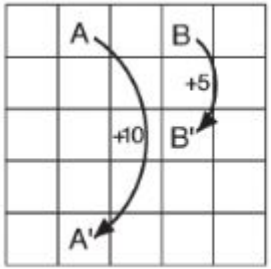


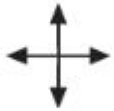


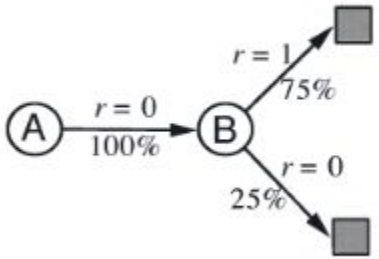
Introducción

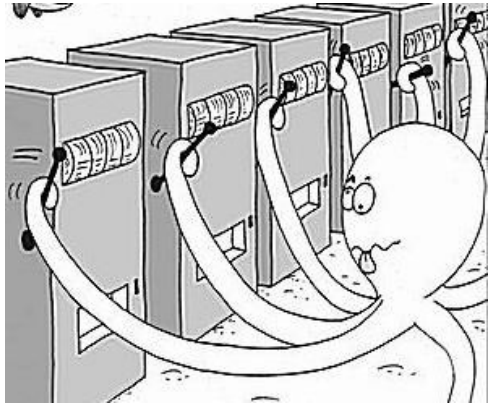
Pre deep-RL (pre-2013)

- Felicidad en mundos de bandidos, grillas y pequeños problemas.
- A RL se le veía mucho potencial pero sin terminar de resolver problemas de interés para la comunidad de ML en general.









Introducción

Pre deep-RL (pre-2013)

- Ejemplo de competencia de 2009, donde una de las posibles alternativas a considerar era RL (junto con algos genéticos, métodos basados en reglas, ...)

Introducción

Boom del deep learning (2012)

RL + deep learning = *deep-RL*

Atari (2013-2015)

Deepmind

Introducción

Boom del deep-RL

AlphaGo (2016)

Deepmind

(ya de Google)

Introducción

Boom del deep-RL

Dota 2 (2018)

OpenAI

Introducción

Boom del deep-RL

AlphaStar (2019)

Deepmind

Introducción

Overhype!!!

Introducción

Paralelamente (~2016), por el *overhype* se crea una brecha entre lo publicitado y la realidad, y surge una crisis de reproducibilidad

- Artículos y presentaciones geniales, pero normalmente no brindan su código, difícil de ejecutarlos o acceder al modelo entrenado.
- Algoritmos dependientes de muchos detalles técnicos no siempre disponibles (no reportados/ocultos/propietarios).

Introducción

Overhype

- Poca ablación, temor a publicar resultados negativos (!).
- A veces no se reporta todo el camino, fallos y obstáculos que llevaron al mejor modelo reportado en los papers (***HARKing***).

Introducción

HARKing - hypothesizing after the results are known

Paper recomendado: HARK Side of Deep Learning - From Grad Student Descent to Automated Machine Learning

- *“Hacer pasar como hipótesis a priori, a una hipótesis formulada tras ver los resultados“*
- *“I view papers through survival bias: if there’s an experiment that’s natural in the paper’s context, but isn’t in the paper, then it probably didn’t work, because if it worked, it’d be in the paper”*

(de https://www.alexirpan.com/2020/05/07/rl-potpourri.html)

Introducción

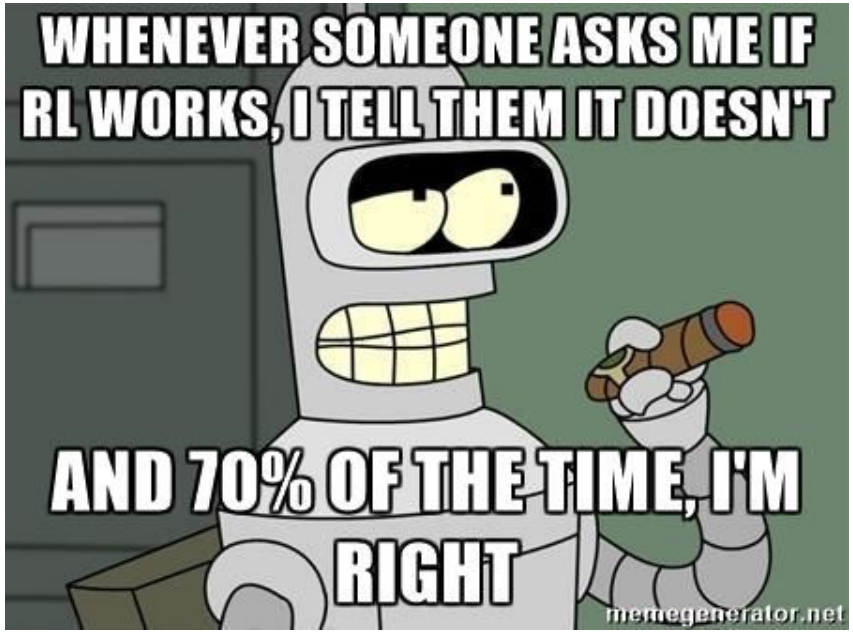
Algunos ejemplos representativos en RL hablando de esto:

- Paper: Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control
- Paper: Deep Reinforcement Learning That Matters
- Paper: Implementation Matters in Deep RL: A Case Study on PPO and TRPO

Introducción

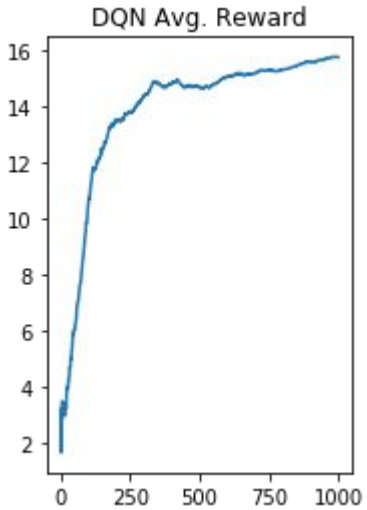
Famoso post de Alex Irpan criticando el estado de deep RL en Febrero 2018

(meme de ese post)



Introducción

- Anécdota de nuestro pequeño intento de hace varios años, de acercarnos al rendimiento de DeepMind en Atari
- Recompensa por DeepMind: ~400 vs. nuestro agente ... ~15 😁



Introducción

Actualidad del RL

- Se van consolidando frameworks de agentes/algoritmos.
- Esto se suma a la API de entornos consolidada en 2016 y entornos comunes.
- RL es **muy flexible** y se beneficia mucho de integrar otras técnicas.
- RL en academia visto como muy promisorio, buena cantidad de investigadores incorporaron RL como foco/parte de sus investigaciones.

Introducción

Actualidad del RL

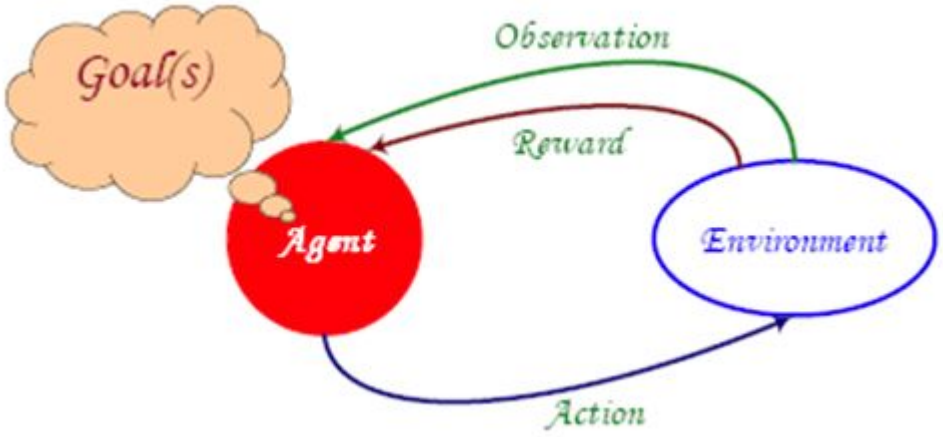
- Algunos desafíos: alta cantidad de muestras necesarias; cómo incorporar mejor el conocimiento de dominio; evitar caer en óptimos locales.
- RL en industria muy usado en su forma más simple (bandidos/tabular), incorporando lentamente soluciones más sofisticadas (próxima clase!).
- Normalmente, en estas aplicaciones RL se usa en conjunto de varias otras técnicas (ej: *imitation learning* o comportamiento basado en reglas)

Nociones básicas

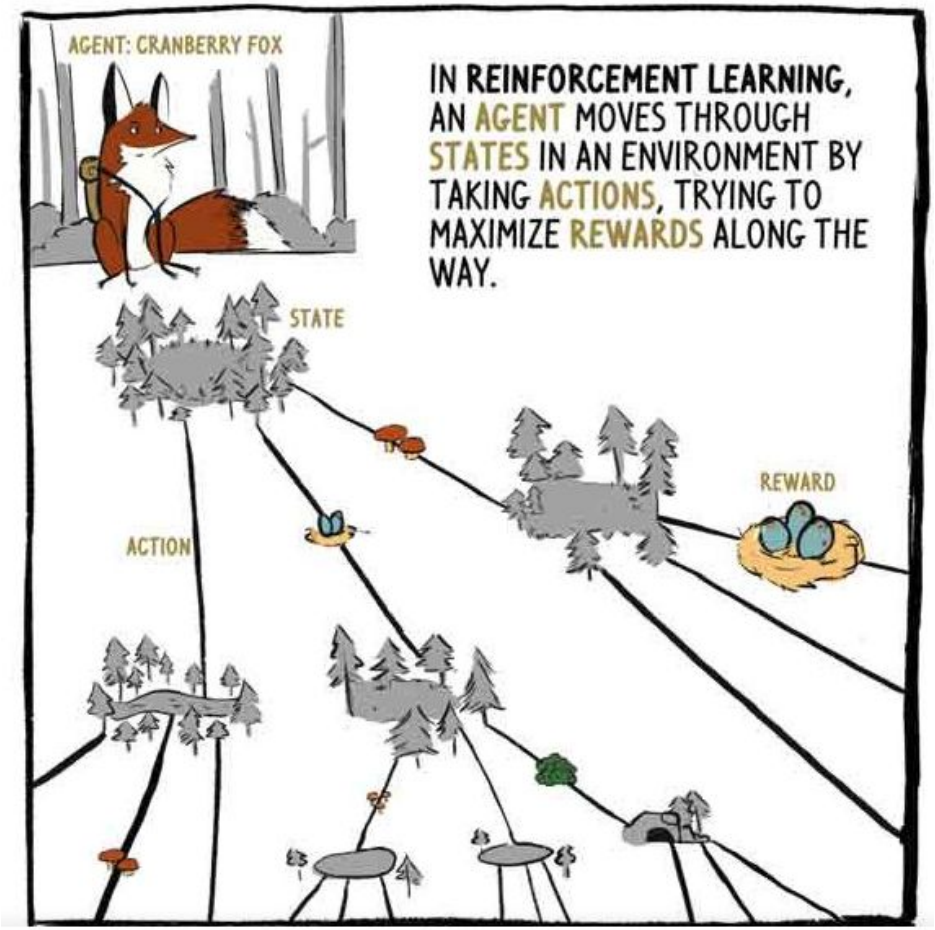
Crédito: Libro *Reinforcement Learning - An Introduction. Sutton & Barto, 2018*

Nociones básicas

Agente buscando un objetivo en su entorno



Nociones básicas

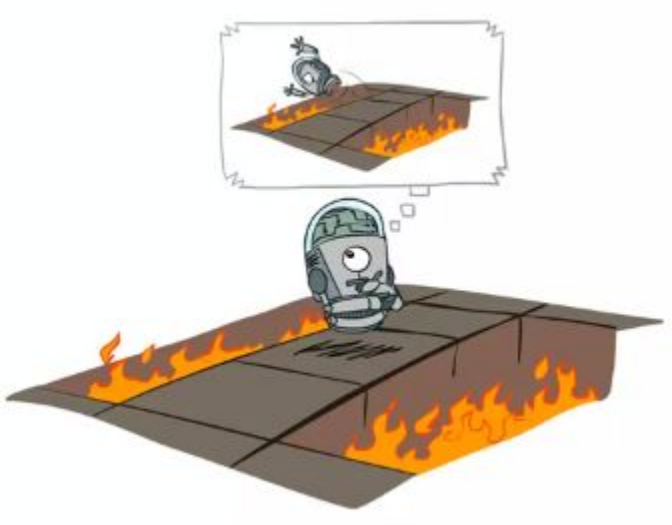


Nociones básicas

- No hay dataset de ejemplos X, **debe generarlos el agente interactuando**.
- Política π, define el comportamiento del agente. Suele expresarse como:
  - $\pi(s)$ devuelve una acción seleccionada bajo la política π.
  - $\pi(a \mid s)$ es la probabilidad de elegir a con la política π.
- ¿Cómo el agente estima las acciones que maximizarán su recompensa?

Nociones básicas

- Dilema de exploración - explotación: ¿cuándo explotar conocimiento vs. cuándo probar nuevas estrategias?



Nociones básicas

Definición formal: 5-upla (S, A, R, P, γ)

- Conjunto de estados S
- Conjunto de acciones A
- Función de recompensa $R(s, a)$
- Función de probabilidad de transición $P(s_{t+1} = s' \mid s, a)$
- Factor de descuento $\gamma \in [0, 1)$

- **Cumple la propiedad de Markov**

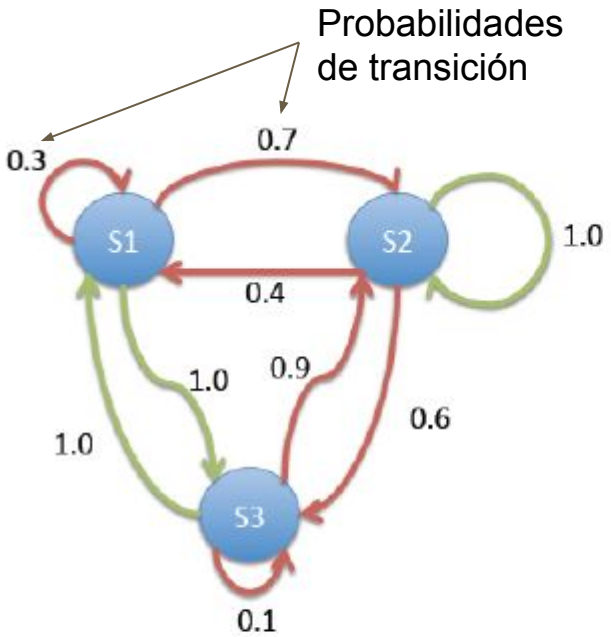
Nociones básicas

Dos acciones:

- Rojo
- Verde

Recompensas:

- R(S1) = 1
- R(S2) = 0
- R(S3) = -1



Función de transición T

s	a	s'	P
S1	R	S1	.3
S1	R	S2	.7
S1	V	S3	1
S2	R	S1	.4
S2	R	S3	.6
S2	V	S2	1
S3	V	S1	1
S3	R	S3	.1
S3	R	S2	.9

Nociones básicas

- Episodio: secuencia $s_0, a_0, r_1, s_1, a_1, s_2, a_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$

  donde s_n es un estado final, (o n es el tiempo de corte)

- Recompensa total del episodio: $R = r_1 + r_2 + \cdots + r_n$

- Recompensa expresada a partir de un instante de tiempo t:

$$R_t = r_t + r_{t+1} + \ldots$$

Nociones básicas

- Aleatoriedad involucrada en los episodios
- ¿Cuán lejos mirar en el horizonte?
- Recompensas acumuladas descontadas por $\gamma \in [0, 1)$

$$
\begin{aligned}
R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots = \\
&= r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots) = \\
&= r_t + \gamma R_{t+1}
\end{aligned}
$$

(la recompensa acumulada descontada también suele denotarse como G_t)

Nociones básicas

- ¿Cómo evalúa el agente lo que vale un estado?
- **Función de valor**

Función de Estado - Valor para la Política π :

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

"La recompensa (descontada) que el agente espera recibir empezando en el estado s y siguiendo la política π"

Nociones básicas

- Forma alternativa: **función de acción-valor *Q* (Q de *quality de a*)**

Función de Acción - Valor para la Política π :

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\}$$

"La recompensa (descontada) que el agente espera recibir empezando en el estado s, tomando la acción a y siguiendo la política π"

Nociones básicas

- Ecuación de Bellman: dado

$$
\begin{aligned}
R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \cdots \\
&= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \cdots \right) \\
&= r_{t+1} + \gamma R_{t+1}
\end{aligned}
$$

- Entonces

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi} \left\{ R_t \middle| s_t = s \right\} \\
&= E_{\pi} \left\{ r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s \right\}
\end{aligned}
$$

“el valor de un estado puede expresarse a partir del siguiente”

Nociones básicas

- Expresado sin el valor esperado

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V^{\pi}(s') \right]$$

" $V(s)$ dado por probabilidad de elegir acción siguiente según la política y el entorno, la probabilidad $P^{a}_{ss'}$ de llegar a ese estado, la recompensa por haber llegado y el valor del estado siguiente"

Nociones básicas

- Optimalidad: si π es óptima π^*, seleccionando siempre la mejor acción, entonces su valor está dado por

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} \boldsymbol{P}(s, a, s') \Big(R(s, a, s') + \gamma V^*(s') \Big)$$

- Dada $V^*(s)$, π^* está dada por

$$\pi^*(s) = \arg\max_{a} \sum_{s' \in S} \boldsymbol{P}(s, a, s') \Big(R(s, a, s') + \gamma V^*(s') \Big)$$

Nociones básicas

- Expresado como función de acción-valor

$$Q^*(s,a) = \sum_{s'} \boldsymbol{P}(s,a,s') \left(R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right)$$

Comparación de políticas de actuación y política óptima

$\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v'_\pi(s)$

$\pi_* \geq \pi' \; \forall \pi'$ *(garantizada)*

Nociones básicas (resumen)

Agente buscando un objetivo en su entorno

Agente:

- F de valor $V(s)$
- Política $\pi(s,a)$
- Modelo del entorno (opc.)

Ambiente (inaccesible para el agente excepto por obs/acción/recomp):

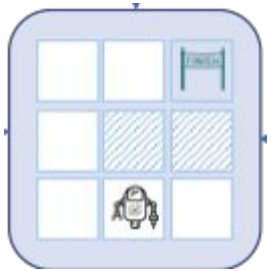
- F de recompensa (oculta) $R(s,a)$
- Prob transición (oculta) $P(s_{t+1} = s' \mid s, a)$

Ejemplos de políticas y MDPs

Crédito: curso Aprendizaje por Refuerzo Profundo, dictado por Juan Gómez Romero en la ECI 2019, Buenos Aires

Ejemplos de políticas y MDPs

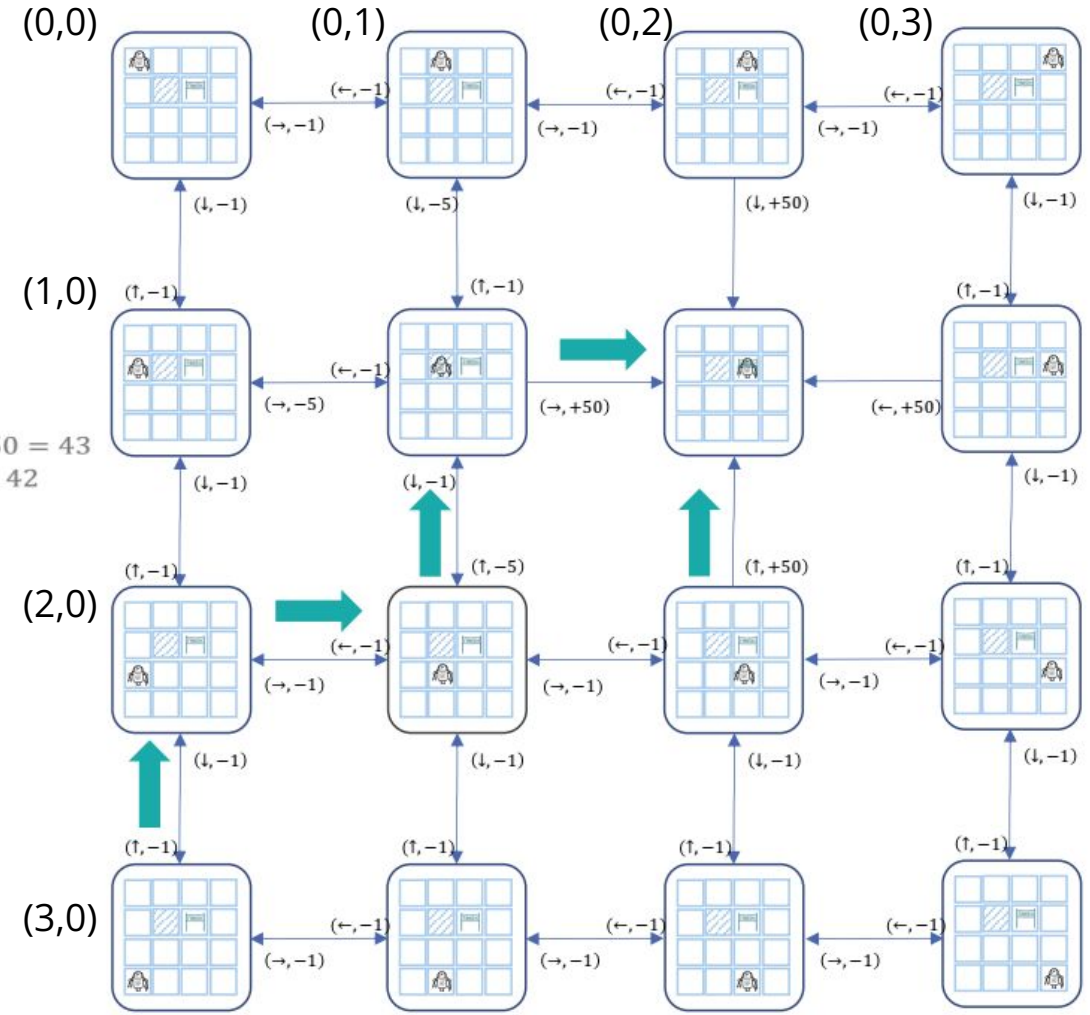
- Robot en una grilla (NxN) intentando llegar a su meta
  - Se mueve arriba, abajo, izquierda, derecha
- La recompensa se basa en la demora en llegar:
  - -1 por cada paso que da
  - -5 al caer al agua (demora más en cruzar y debe secarse)
- Llegar a la meta le da +50 recompensa



Política π_1 $(\gamma = 1)$:

Estado	Acción
$(3,0)$	↑
$(2,0)$	→
$(2,1)$	↑
$(1,1)$	→
$(2,2)$	↑
...	

$$v_{\pi_1}(\langle 2,1\rangle) = -5 + 50 = 45$$
$$v_{\pi_1}(\langle 2,2\rangle) = 50$$
$$v_{\pi_1}(\langle 3,0\rangle) = -1 - 1 - 5 + 50 = 43$$
$$v_{\pi_1}(\langle 2,0\rangle) = -1 - 5 + 50 = 42$$



Política π_2 $(\gamma = 1)$:

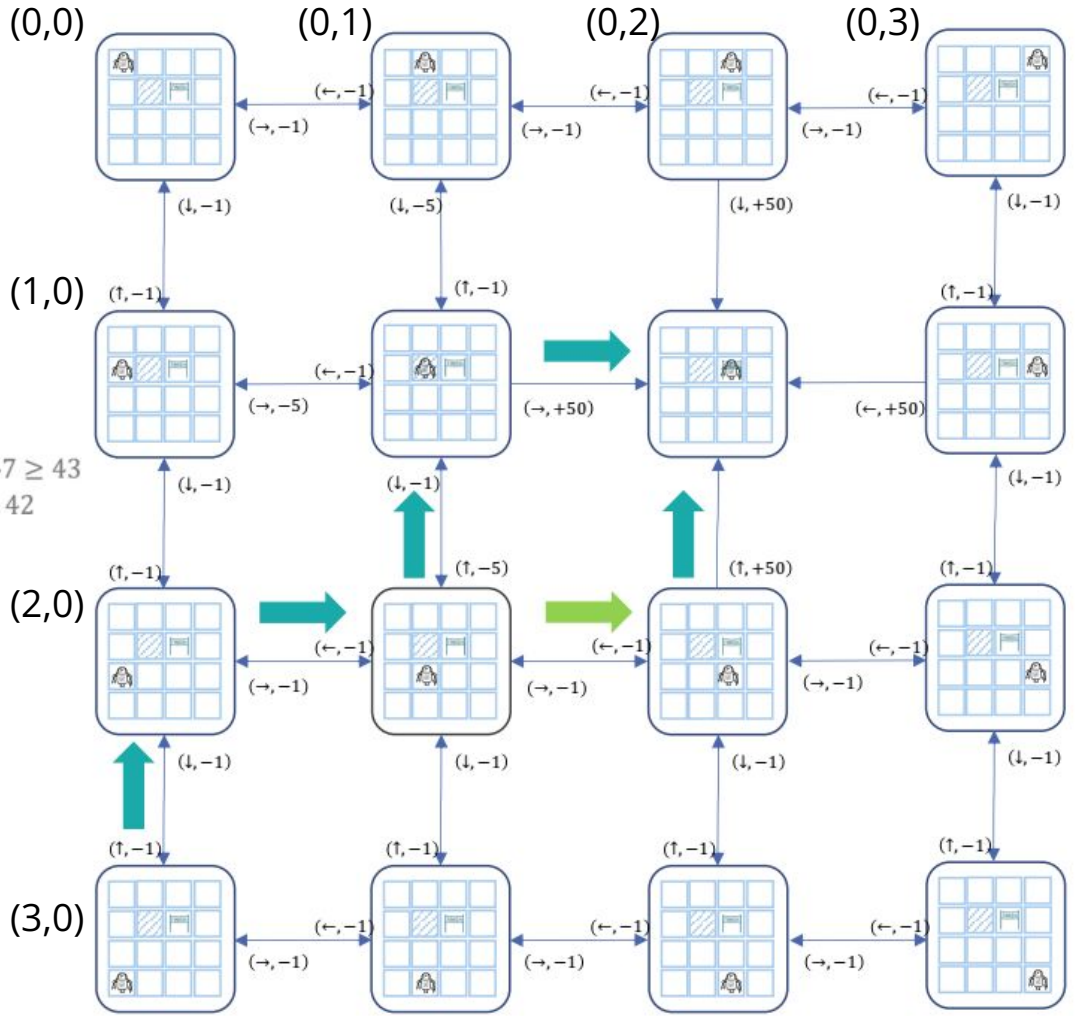
Estado	Acción
$(3,0)$	↑
$(2,0)$	→
$(2,1)$	→
$(1,1)$	→
$(2,2)$	↑
...	

$v_{\pi_2}(\langle 2,1\rangle) = -1 + 50 = 49 \geq 45$

$v_{\pi_2}(\langle 2,2\rangle) = 50 \geq 50$

$v_{\pi_2}(\langle 3,0\rangle) = -1 - 1 - 1 + 50 = 47 \geq 43$

$v_{\pi_2}(\langle 2,0\rangle) = -1 - 1 + 50 = 48 \geq 42$



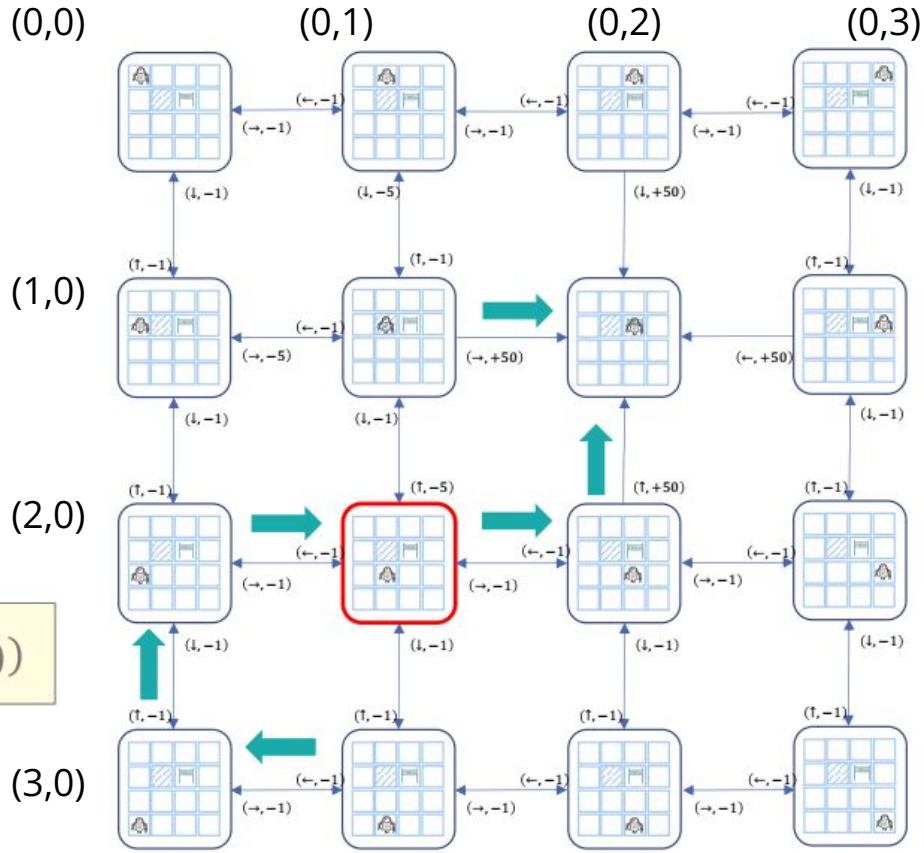
$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

Política π_2 $(\gamma = 1)$:

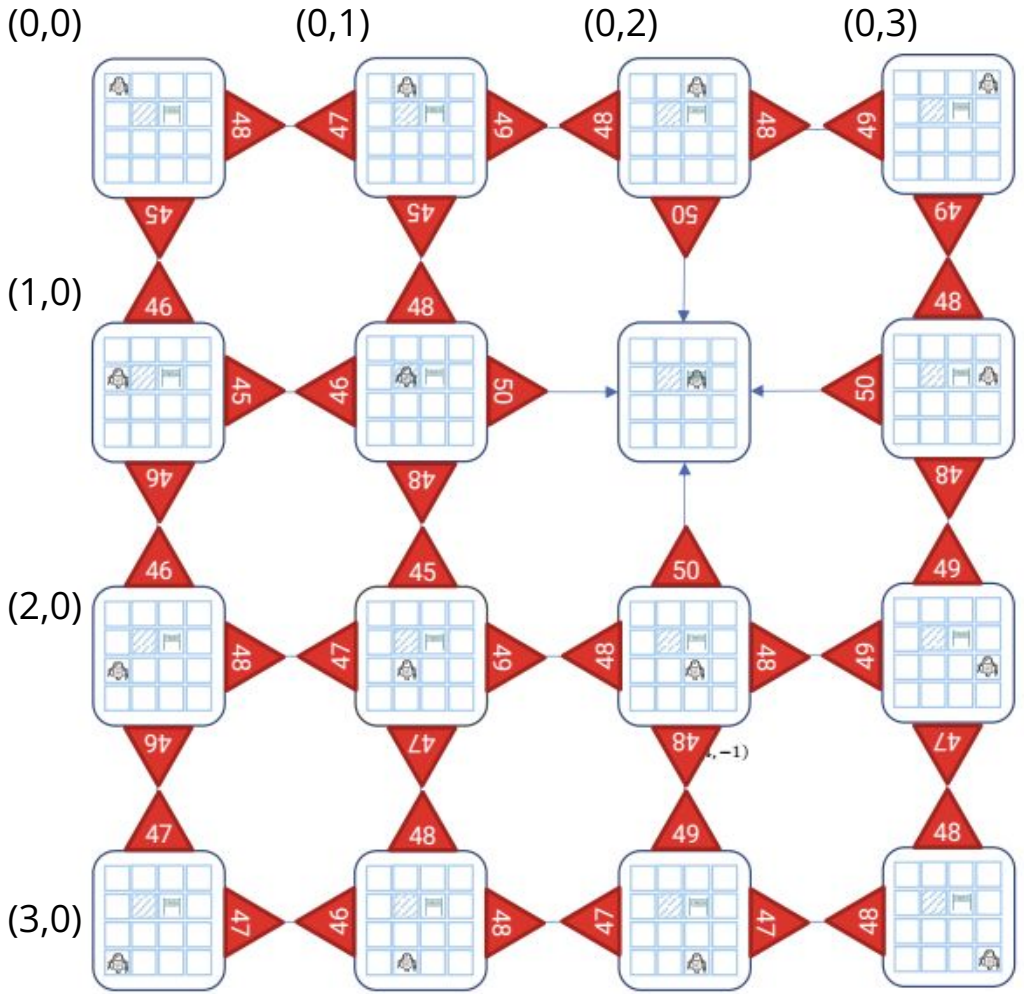
Estado	Acción
$(3,0)$	$\uparrow$
$(2,0)$	$\rightarrow$
$(2,1)$	$\rightarrow$
$(1,1)$	$\rightarrow$
$(2,2)$	$\uparrow$
$(3,1)$	$\leftarrow$
...	

$$v_\pi(s) = q_\pi(s, \pi(a|s))$$

$$q_{\pi_2}(\langle 2,1\rangle, \uparrow) = -5 + 50 = 45$$
$$q_{\pi_2}(\langle 2,1\rangle, \rightarrow) = -1 + 50 = \mathbf{49}$$
$$q_{\pi_2}(\langle 2,1\rangle, \leftarrow) = -1 - 1 - 1 + 50 = 47$$
$$q_{\pi_2}(\langle 2,1\rangle, \downarrow) = -1 - 1 - 1 - 1 - 1 + 50 = 45$$



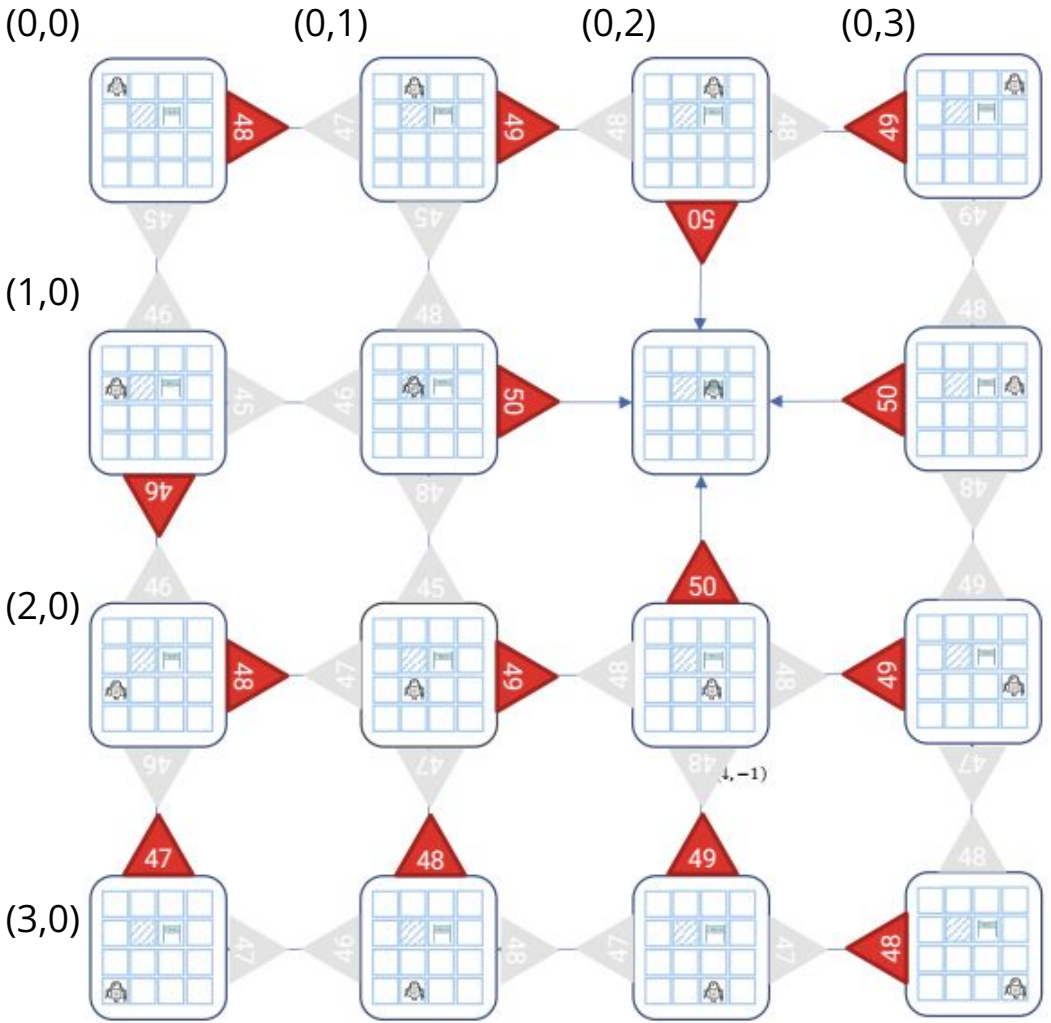
$q_{\pi_*}(s, a)$



$$q_{\pi_*}(s,a)$$

Resaltar valor máximo de cada estado:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a)$$



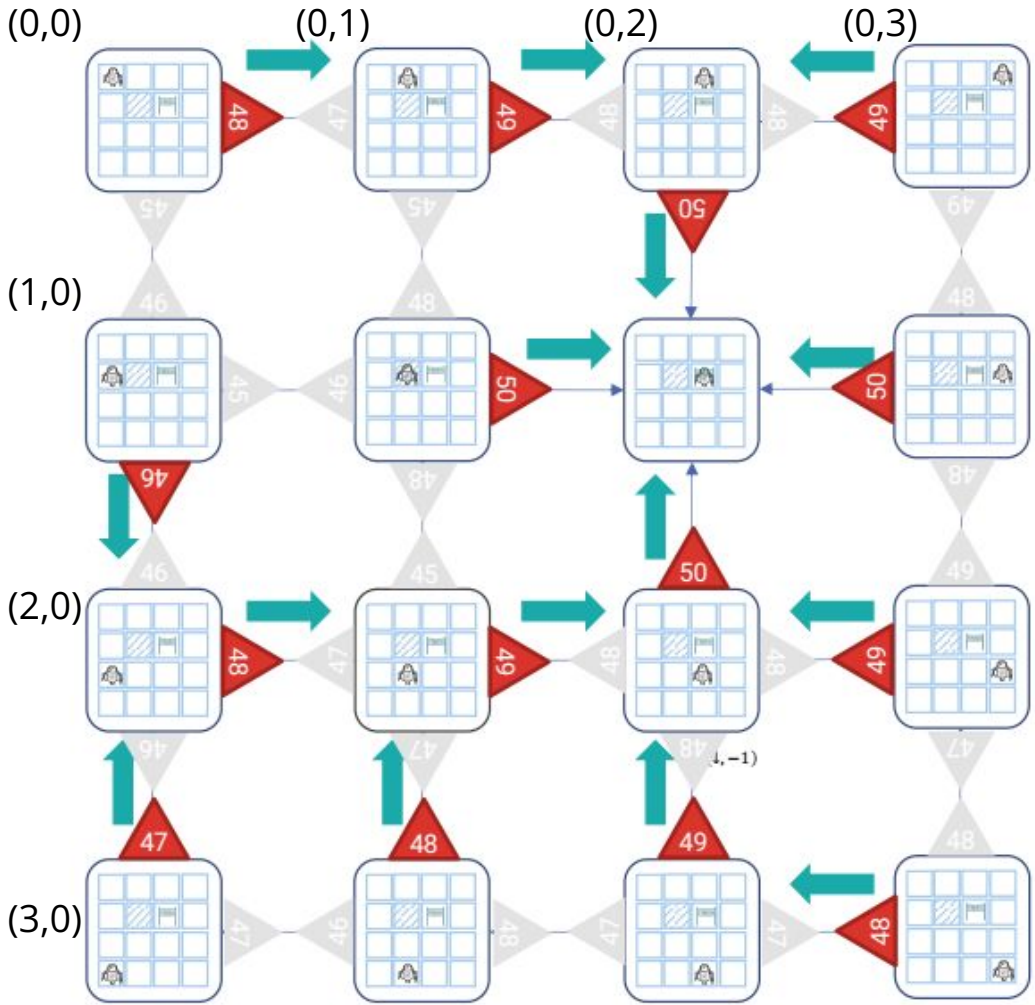
$q_{\pi_*}(s,a)$

Resaltar valor máximo de cada estado:

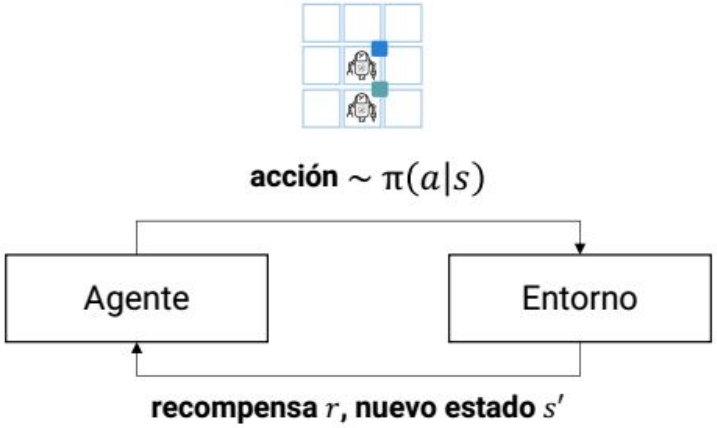
$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a)$$

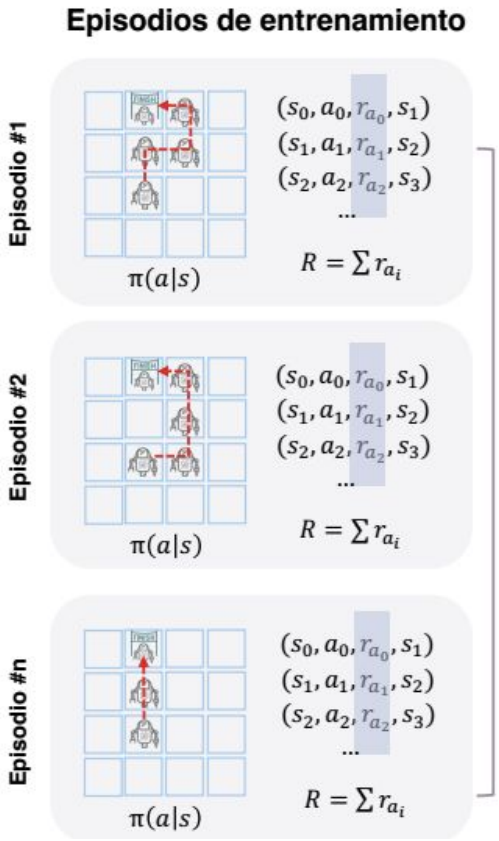
Reconstruir π_*

Estado	Acción
$(3,0)$	→
$(2,0)$	↑
...	



¿Cómo llegamos a esta política? con entrenamiento (próximas slides!)





Estado	Acción	q
(3,0)	↑	10
(2,0)	→	12
...		...

training

$$\pi' = \arg\max_{\pi} \sum_{t=0} \gamma^t * r_{a_t}(s_t, s_{t+1})$$

con $a_t \sim \pi(a|s)$

$\gamma \in [0,1]$: tasa de descuento

Actividad 0: ejercicios a mano

Link al notebook del Lab 0

Formas de resolver RL

Algunas formas básicas de resolver RL

- Bandidos (a continuación)
- Programación dinámica:
  Sabemos $P(s_{t+1} = s' \mid s, a)$ y $R(s, a)$, calculamos/aproximamos $V(s)$

- Métodos de Monte-Carlo
  Actualizamos $V(s)$ tras juntar recompensas de cada episodio

- Diferencia temporal (a continuación)

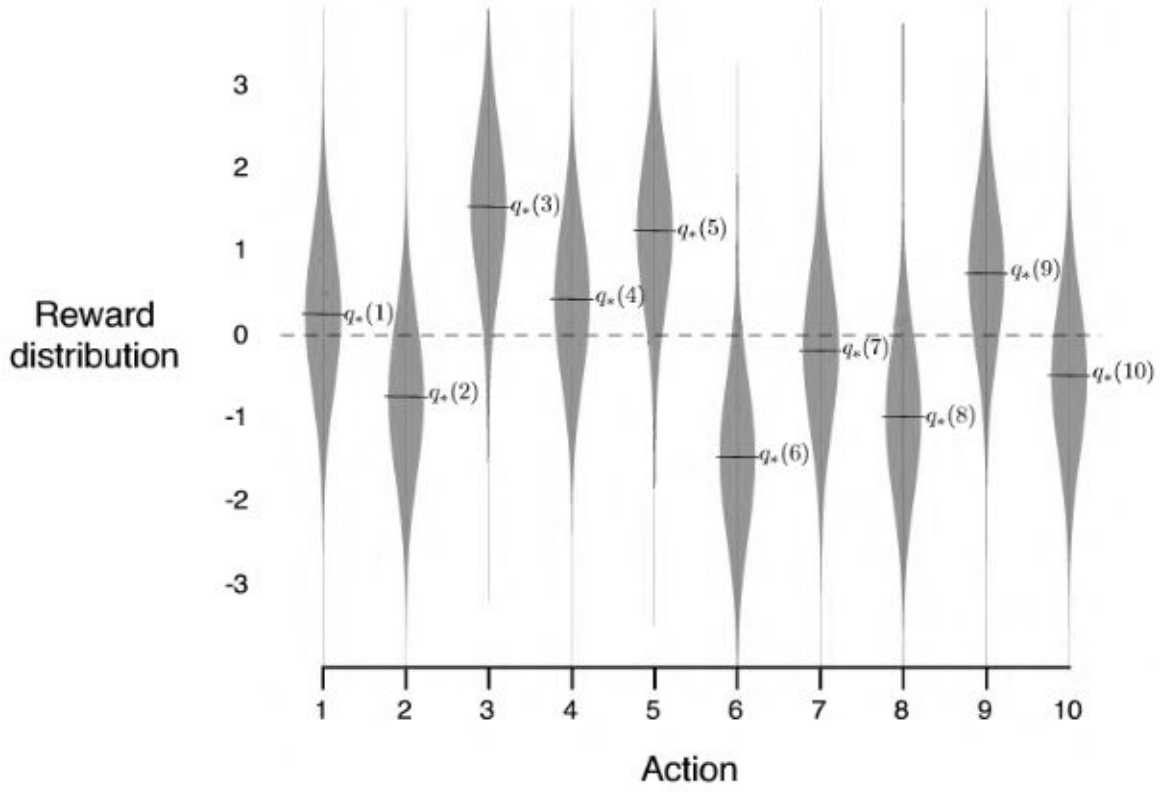
- Aproximación de política:
  Calculamos $\pi(s)$ mediante *ascenso* de gradiente (próximo fin de semana)

Bandidos

Bandidos

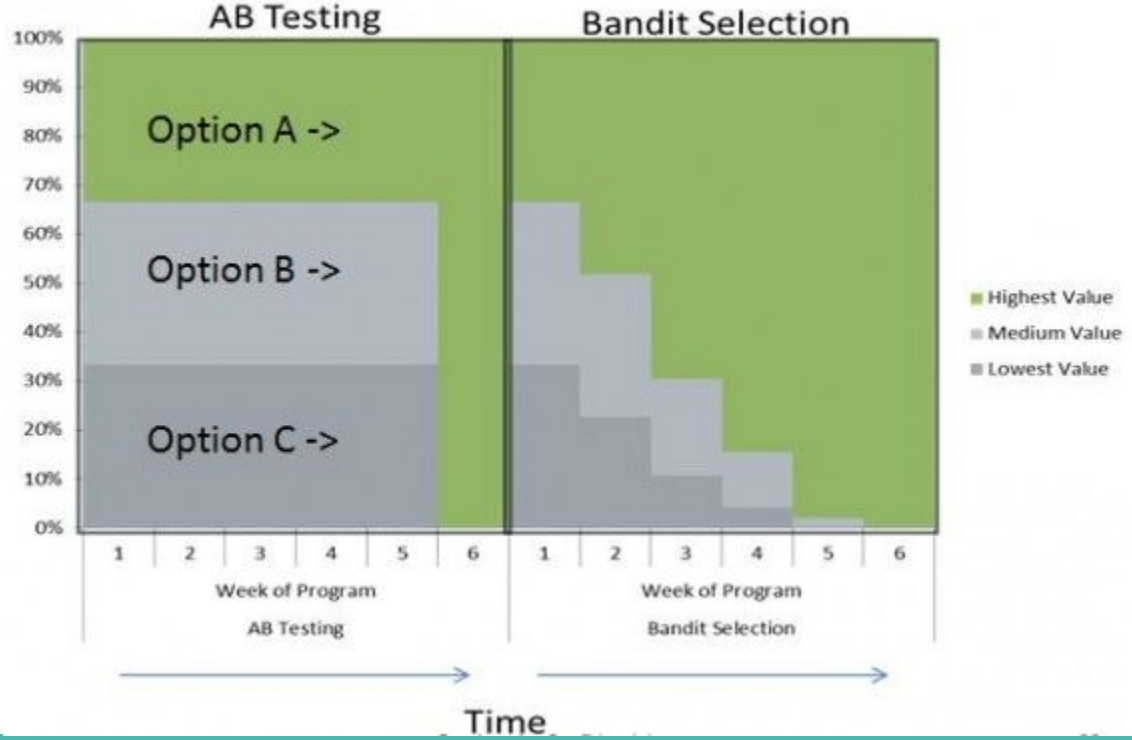
Bandidos, o RL de un sólo estado, donde cada paso es un episodio

Bandidos


Reward distribution
3
2
1
0
-1
-2
-3
$q_*(1)$
$q_*(2)$
$q_*(3)$
$q_*(4)$
$q_*(5)$
$q_*(6)$
$q_*(7)$
$q_*(8)$
$q_*(9)$
$q_*(10)$
1
2
3
4
5
6
7
8
9
10
Action

Bandidos

- Alternativa a A/B Testing



Fuente

Bandidos

- A veces incluyen un contexto, son muy usados de esta forma.
- Spotify, por ejemplo, usa bandidos contextuales para hacer recomendaciones
- Ej: ¿qué recomendación funciona mejor, para una persona que le gusta escuchar Jazz?

Bandidos

- Los bandidos suelen usarse en su versión más simple, o como *bandidos contextuales*.
- Buen repo con librería de bandidos + links para aprender más.

Bandidos - bases del RL

Los bandidos sientan varias de las bases del RL, tienen heurísticas de selección comunes a otros métodos:

ε-greedy: estando en estado s, elegir la acción con mayor Q con probabilidad

$$\pi(a \mid s) = 1 - \varepsilon$$

- o bien -

elegir una acción aleatoria con probabilidad ε

Bandidos - bases del RL

Otra forma: la probabilidad de la acción a elegir se pesa por la magnitud de su Q

Entonces, para cada acción a, su probabilidad de ser elegida pasa a tomar una forma de este estilo...

$$\pi(a \mid s) = \frac{Q(s,a)}{\sum_{\tilde{a} \in A} Q(s,\tilde{a})}$$

Bandidos - bases del RL

La forma más común de esto es mediante la función Softmax:

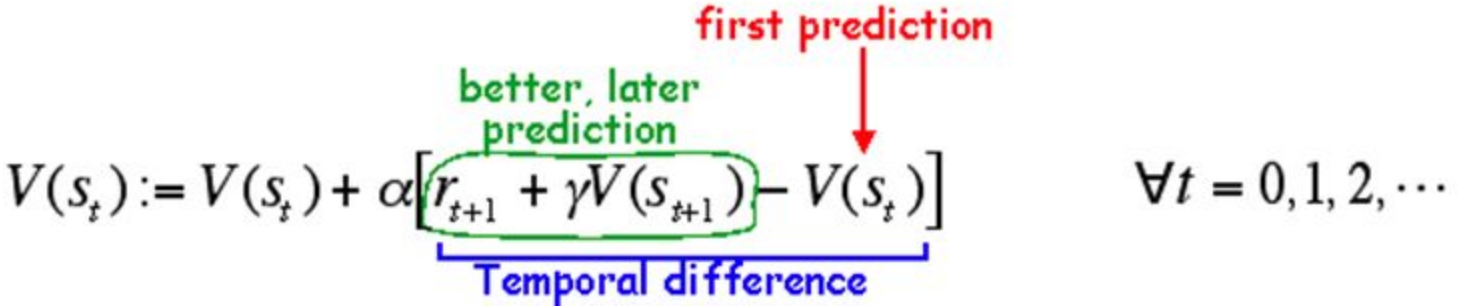
$$\pi(a \mid s) = \frac{e^{Q(s,a)/\tau}}{\sum_{\tilde{a} \in A} e^{Q(s,\tilde{a})/\tau}}$$

donde τ es la “temperatura computacional” (+ temp. -> + exploración)

Diferencia Temporal (TD)

Diferencia temporal (TD)

- Actualizar una predicción de $V(s)$ en base al cambio que existe en la misma de un momento (*t*) al siguiente (*t+1*)



Diferencia temporal (TD)

- Los algoritmos de aprendizaje basados en TD se emplean en mayor medida para realizar el **control** respecto de las acciones que ejecuta un agente que interactúa con su entorno.

- Así, en lugar de aprender la función de estado-valor $V(s)$, se orientan al aprendizaje de la función de acción-valor $Q(s, a)$

Diferencia temporal (TD)

Enfoques principales para realizar el aprendizaje de funciones $Q(s, a)$

On-policy y off-policy:

- On-policy: estima $Q(s, a)$ para la **política** π siendo ejecutada
- Off-policy: estima $Q(s, a)$ para la **función óptima** de acción-valor, Q^*

Diferencia temporal (TD)

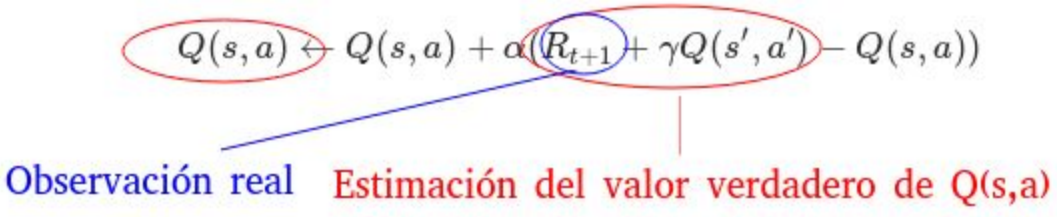
- On-policy

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

- s_t, a_t seleccionados a partir de la política π (es decir que **son ejecutados en el ambiente**)

Diferencia temporal (TD)

- Bootstrapping: "emplear uno o más valores estimados en la actualización del mismo tipo de valor estimado"



Diferencia temporal (TD)

- Algoritmo *SARSA (State - Action - Reward - State - Action)*

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., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Diferencia temporal (TD)

- Off-policy

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \arg\max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$$

- Es decir que se busca aproximar directamente Q^* (política óptima, por eso es off-policy)

Diferencia temporal (TD)

- Algoritmo Q-Learning

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

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
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Métodos basados en modelos

Métodos basados en modelos

- Hasta recién, vimos algoritmos *“model-free”*, que es una forma muy común de usar agentes.
- Estos métodos basan totalmente su aprendizaje en la interacción directa con el entorno, tomando mínimas asunciones.
- ¿y si intententamos modelar aspectos desconocidos como R(.) y P(.)?

Métodos basados en modelos (*Model-based RL)*

Métodos basados en modelos

- Un modelo del entorno representa cualquier elemento que permita al agente predecir cómo el entorno responderá ante las acciones que ejecuta.
- Dado un estado y una acción, el modelo produce una predicción del siguiente estado y la recompensa asociada.
- Si el entorno es estocástico, entonces existen distintos posibles estados siguientes/recompensas, cada uno con distinta probabilidad de ocurrir.

Métodos basados en modelos

- Algunos modelos producen una descripción de todas las posibilidades con sus probabilidades asociadas: **modelos de distribución**.
- Estos generan todas las posibles transiciones, pesadas por su probabilidad de ocurrir.
- Esto permitiría, dado un estado-acción inicial, producir todos los posibles episodios y posibilidades!

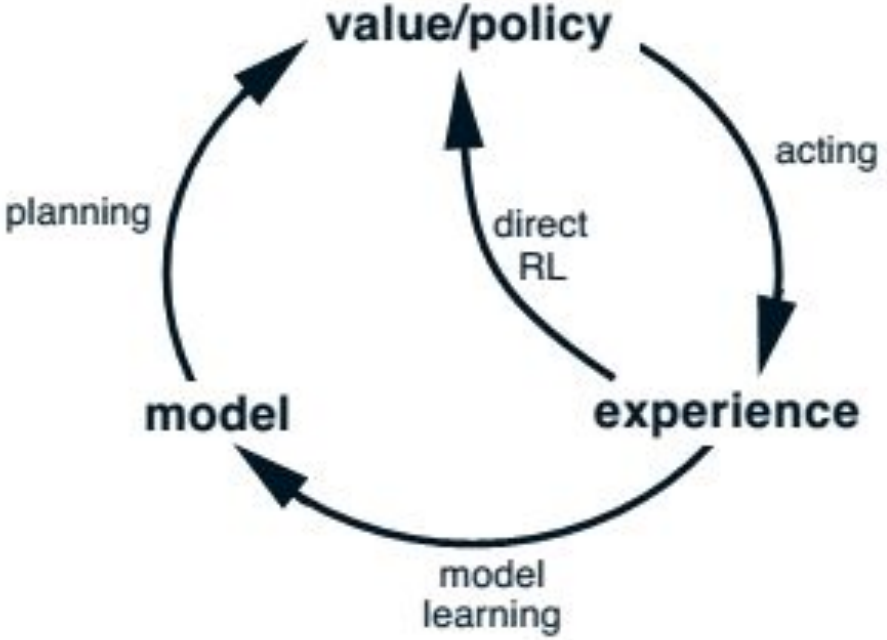
Métodos basados en modelos

- Otros modelos producen sólo una de las posibilidades, muestreando las mismas en función de su probabilidad según los ejemplos obtenidos: los **modelos basados en ejemplos**.
- Dado un estado-acción inicial, esto permitiría producir un episodio completo.
- Ambos tipos de modelos pueden ser empleados para replicar o generar experiencia.

Métodos basados en modelos

- **Planning** en RL se utiliza para referirse a cualquier proceso computacional que emplea el modelo como entrada y produce o mejora una política de interacción con un entorno determinado.
- El planning puede ser realizado de forma on-line y off-line.

Métodos basados en modelos



Métodos basados en modelos

- Cuando el planning es realizado *on-line*, mientras se interactúa con un entorno, surgen cuestiones a resolver:
- Por ejemplo, la nueva información proveniente del entorno podría impactar en el modelo y de esa manera modificar el proceso de planning.
- ¿Cómo dividir los recursos entre la toma de decisiones y el aprendizaje de modelos (*decision-making* y *model-learning* respectivamente)?

Métodos basados en modelos - Dyna-Q

- Dyna-Q es un algoritmo clásico que integra las funciones más importantes para un agente que realiza planificación (planning) on-line.
- La misma, establece al menos dos roles para la experiencia real generada por el agente:
  1. Mejorar el modelo (model learning, hacer que refleje la realidad lo mejor posible)
  2. Mejorar la función de valor (y, por ende, la política), empleando los métodos basados en Aprendizaje por Refuerzo (direct-RL).

Métodos basados en modelos - Dyna-Q

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:

(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:

$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Métodos basados en modelos

¿Qué sucede con nuestro modelo si el entorno cambia?

- En muchos entornos determinísticos, una vez aprendido el modelo, se puede usar de manera directa para propagar cambios producidos por las diferencias temporales.
- En entornos no determinísticos, o en aquellos en los cuales se de el hecho de que en algún momento de la interacción agente-entorno este último cambia, es necesario incorporar mecanismos de auto-corrección de modelos para evitar obtener políticas sub-óptimas.

Métodos basados en modelos

- El problema general de corrección de modelos es otra versión del dilema de exploración-explotación.
- En el contexto del planning en RL, la exploración significa ejecutar acciones que mejoran el modelo; la explotación significa comportarse de manera óptima de acuerdo a lo establecido por el modelo.
- Es decir, se busca que el agente explore para detectar cambios en el entorno, pero sin degradar la performance del mismo de manera determinante.

Métodos basados en modelos - Dyna-Q+

- El método Dyna-Q+ propone resolver esto empleando una heurística basada en la cantidad de tiempo que un par estado-acción no es visitado de manera real por el agente.
- Mientras más tiempo haya pasado desde una interacción real, mayor es la probabilidad de que las dinámicas en determinados pares estado-acción hayan cambiado, (y el modelo sea incorrecto).

Métodos basados en modelos - Dyna-Q+

- Para motivar el testeo de acciones que hace largo tiempo no se ejecutan en determinado estado, se otorga un bonus especial a las acciones simuladas que tienen en cuenta dichas acciones.
- En particular, si la recompensa para una transición determinada es R, y dicha transición no ha sido ejecutada en τ time steps, entonces las actualizaciones basadas en planning se hacen como si la misma produjera una recompensa de $R + \kappa\sqrt{\tau}$.
- El parámetro κ se denomina “curiosidad computacional”.

Métodos basados en modelos

Ventajas de estos métodos:

- Permiten reducir la interacción directa con el entorno (cuando muestrearlos puede ser costoso, o puede no ser siempre posible).
- Permiten modelar aspectos internos del entorno de forma más directa que sólo utilizando el valor (y complementarse con él!).

A tener en cuenta:

- Agregan asunciones e hiper-parámetros.
- Pueden ser intensivos computacionalmente.

Ética en RL

Ética en RL

- RL involucra agentes que toman decisiones, buscando agresivamente un objetivo
- Es muy importante tener en cuenta las posibles implicaciones éticas, en particular porque estamos maximizando una métrica (recompensa) que si no toma en cuenta los posibles desenlaces, puede tener nefastas consecuencias.
- Una de las áreas de estudio reciente en esta dirección es el **safe reinforcement learning**.

Ética en RL

"Nope, I gotta explore" — Stupid RL agent

Fuente de la imagen

Ética en RL

- Por contrapartida, RL se usa en aplicaciones como aviones de combate y misiles, por ejemplo ...
- Tesis de maestría: Reinforcement learning applications to combat identification

Ética en RL

- Papers:
  Maneuver Decision of UAV in Short-Range Air Combat Based on Deep Reinforcement Learning (*UAV = drone*)

  A Deep Reinforcement Learning Based Intelligent Decision Method for UCAV Air Combat (*UCAV = combat drone*)

  Efficient Training Techniques for Multi-Agent Reinforcement Learning in Combat Tasks

Ética en RL

- Más papers:
  An Empirical Study of Reward Structures for Actor-Critic Reinforcement Learning in Air Combat Manoeuvring Simulation

  Autonomous Control of Combat Unmanned Aerial Vehicles to Evade Surface-to-Air Missiles Using Deep Reinforcement Learning

  Agent Coordination in Air Combat Simulation using Multi-Agent Deep Reinforcement Learning

Actividad 1: Programar primeros agentes

Link al notebook del Lab 1