Aprendizaje por refuerzo

Aprenentatge Automàtic

APA/GEI/FIB/UPC - 2025/2026 1Q

/ Javier Béjar

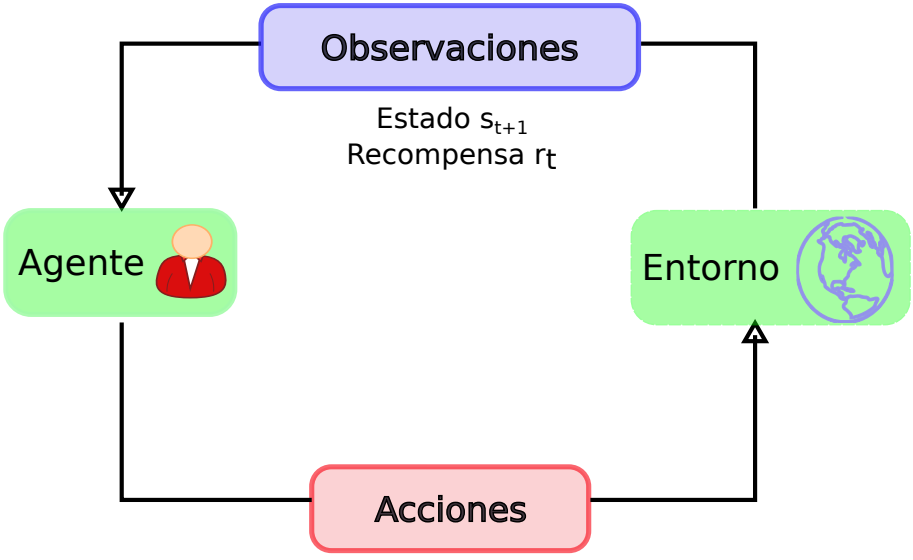
Introducción

- Un agente debe resolver una tarea y tiene un conjunto de acciones disponibles
- El agente elige la acción o conjunto de acciones que cree que pueden resolver la tarea
- El agente recibe información de retroalimentación del entorno sobre su desempeño
- El agente usa esta información para modificar su comportamiento

- El aprendizaje por refuerzo se usa cuando un agente puede interactuar con el entorno
- El entorno es capaz de cuantificar el éxito o el fracaso de las acciones del agente
- Los problemas pueden tener diferentes complejidades
  - Espacio de estados discreto o continuo
  - Acciones discretas o continuas
  - Estados o efectos de las acciones desconocidos
  - Efectos no deterministas de las acciones
  - Refuerzos al final de la tarea o a cada paso

- Se usa para problemas de decisión demasiado complejos para ser resueltos codificados a mano
  - Demasiadas acciones/estados, es difícil vincular las acciones a los objetivos
- Es un enfoque más fácil permitir que el agente aprenda la tarea por ensayo y error
- Aplicaciones: robótica, juegos de computadora, planificación de acciones, resolución/razonamiento de problemas, toma de decisiones, generación de secuencias...

- Un conjunto de estados que describen el entorno (discreto o continuo)
- Un conjunto de acciones que puede realizar el agente para modificar el entorno
- Una función de refuerzo que informa al agente del resultado obtenido al realizar la acción sobre el entorno
- **Objetivo:** Encontrar una política que vincula estados con acciones y maximiza las ganancias obtenidas de los refuerzos


Observaciones
Estado s_{t+1}
Recompensa r_t
Agente
Entorno
Acciones
Acción a_t

- El objetivo de aprendizaje de un agente está definido por la retroalimentación del entorno, la señal que llamamos recompensa
- No tiene sentido usar solo la recompensa inmediata, ya que al movemos de un estado a otro cambia la distribución de lo que podemos conseguir posteriormente
- Para aprender el objetivo debemos maximizar la recompensa acumulativa futura, la que se obtiene de los pasos futuros
- Esta recompensa permite optimizar decisiones que solo se pueden optimizar con información de largo plazo (ej. ¿por qué es bueno reciclar?)

- Para tareas episódicas, la recompensa acumulativa de elegir una acción después del tiempo t se define como:

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

T es el momento en el que se alcanza el estado final

- Para tareas continuas (sin estados absorbentes) T podría ser infinito, por lo que en su lugar se considera recompensa con descuento ($0 \leq \gamma \leq 1$):

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Podemos unificar estos dos problemas usando la definición:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

  incluyendo las posibilidades de $T = \infty$ o $\gamma = 1$ (pero no ambas)

- El valor de γ representa la influencia del futuro en la decisión actual

Procesos de decisión de Markov

- Hay varios escenarios posibles para el aprendizaje por refuerzo
- Vamos a suponer que:
  - El estado se puede observar (mediante los sensores del agente)
  - El entorno proporciona una función de recompensa
  - Tenemos un conjunto de acciones para cambiar el estado
  - La tarea tiene la propiedad de Markov (las decisiones futuras no dependen del pasado)
- Estos problemas se llaman Procesos de decisión de Markov (MDP)
- Estos problemas se pueden definir por sus estados, acciones y una dinámica de un paso

- Un agente puede percibir un conjunto S de estados posibles
- Hay un conjunto A de acciones que se pueden realizar
- Cada intervalo de tiempo (discreto) el agente observa el estado actual (s_t) y elige una acción para realizar (a_t)
- El entorno otorga al agente una recompensa correspondiente al estado y la acción ($r_t = r(s_t, a_t)$) y cambia al estado siguiente ($s_{t+1} = \delta(s_t, a_t)$)
- r y δ son funciones del entorno y no son necesariamente conocidas por el agente

- Una política es un conjunto de acciones a realizar para resolver un problema
  - Por ejemplo: La secuencia de movimientos en un juego
- Para un problema, aprenderemos el mejor política (π) donde $\pi(s, a)$ representará la probabilidad de realizar una acción a desde el estado s
- Para estimar una política podemos usar dos aproximaciones
  1. Aproximar la función que estima la recompensa a futuro de un estado (Value learning)
  2. Aproximar la función que estima la distribución de las acciones desde un estado (Policy learning)
- Estas funciones se definen en términos de la recompensa esperada futura

Value learning

- Queremos una función $Q(s, a)$ capaz de predecir cuanto refuerzo obtendremos si usamos una acción desde este estado
- Las acciones a realizar las obtendremos como:

$$a = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Policy learning

- Queremos una función $\pi(s)$ que represente la distribución de las acciones desde un estado
- Las acciones a realizar las obtendremos como:

$$muestra \;\; a \sim \pi(s)$$

Value Learning

- El Aprendizaje por diferencias temporales usa el rendimiento obtenido después de un número limitado de pasos
- El método de aprendizaje por diferencias temporales más simple es Q-learning que usa la estimación del refuerzo a un paso
- Este método no necesita tener un modelo de la tarea
- Se puede implementar en línea dado que no necesita un estado final para obtener la recompensa final (utiliza las estimaciones)

- El objetivo de Q-learning es aproximar el valor de una acción desde un estado (Q) al observar las recompensas obtenidas durante la realización de la tarea
- El modelo consiste en una tabla de $estados \times acciones$ que almacena y actualiza las estimaciones durante el entrenamiento
- Cada paso la función Q se actualiza con
  - La recompensa inmediata r_t otorgada por la acción realizada
  - La diferencia entre la estimación de la mejor acción a para el siguiente estado s_{t+1} y la estimación de la recompensa actual

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

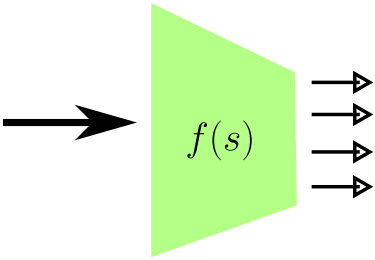
- γ es el descuento y α la influencia del futuro en la actualización

- En problemas con muchos estados es más sencillo el aproximar esta función con una red neuronal
- En este caso la red producirá un valor para cada acción posible (regresión múltiple)
- La política la obtendremos igual, escogiendo el valor máximo
- El entrenamiento usará una función de pérdida que minimice la diferencia entre la recompensa observada y la estimada

$$Q_{loss} = ||[r_t + \gamma \max_a Q(s_{t+1}, a)] - Q(s_t, a_t)||^2$$

- Una ventaja adicional es que cada ajuste afecta a toda la función, aprendemos más rápido

Estado (s)



$Q(s, a_1)$
$Q(s, a_2)$
$Q(s, a_3)$
$\vdots$
$Q(s, a_n)$

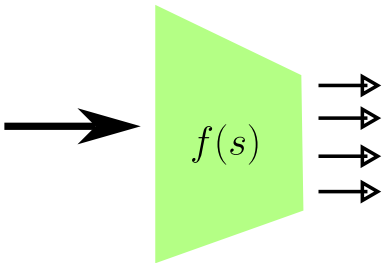
arg max Q(s,a)

- Solo permite aprender en problemas con un número de acciones discreto y pequeño
- No permite dominios con acciones continuas
- La política que obtenemos es determinista (elegir siempre la acción que maximiza recompensa)
- No podemos aprender políticas estocásticas o trabajar en dominios donde el entorno es estocástico

Policy learning

- El objetivo es aprender la función $\pi(s)$ que representa la distribución de probabilidad de las acciones a partir de un estado
- Esta distribución puede ser:
  - Discreta: probabilidad sobre un conjunto de acciones finito
  - Continua: probabilidad sobre los posibles valores que puede tomar una acción o acciones
- Esto permite aprender en dominios con acciones continuas simplemente definiendo que distribución continua siguen sus valores
- También podemos aproximar la función con una red neuronal

Estado (s)



$$p(a_1|s)$$
$$p(a_2|s)$$
$$\vdots$$
$$p(a_n|s)$$

$$\pi(s) \sim p(a|s) = a_3$$

Estado (s)

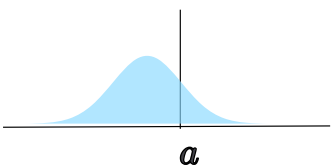
$f(s)$

$\mu = -0.3$

$\sigma^2 = 1.1$

$P(a|s) = \mathcal{N}(\mu, \sigma^2)$

$\pi(s) \sim p(a|s) = -0.4$



1. Inicializar el agente en un estado
2. Seguir la política actual hasta llegar a un estado final
3. Registrar todos los estados, acciones y recompensas obtenidas
4. Aumentar la probabilidad de las acciones con alta recompensa
5. Reducir la probabilidad de las acciones con baja recompensa

- La función que estamos aprendiendo corresponde a una distribución de probabilidad
- Podemos usar como función de pérdida la log verosimilitud
- Esta verosimilitud estará ponderada por los refuerzos observados

$$\ell = \sum_t -\log(p(a_t|s_t)) \cdot r_t$$

- Los pesos de la red se actualizarán a partir del gradiente de esta función

Aquí tenéis varias demos de aplicación de Q-learning y policy learning a dominios sencillos discretos y continuos