Aprendizaje Automático

Introducción a Aprendizaje por Refuerzos

Viviana Cotik

1er cuatrimestre 2019

Tipos de Aprendizaje Automático

- Supervisado
- No supervisado
- **Por refuerzos**
  - **no tiene entrenador. Interactúa con el ambiente y tiene premios y castigos.**
  - **Tarea: aprende a elegir acciones óptimas para lograr su objetivo**

Aplicaciones

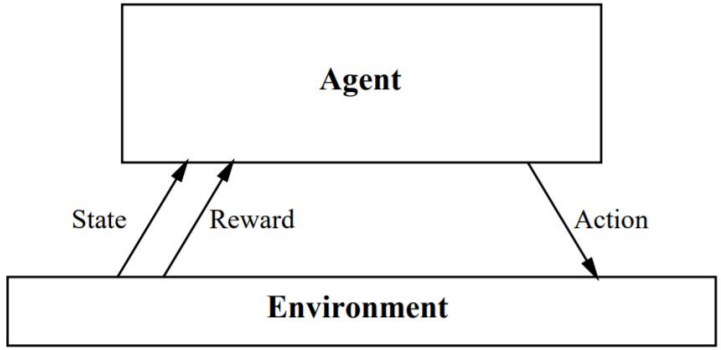
Juegos: backgammon, go

Robots autónomos: https://www.youtube.com/watch?v=0JL04JJjocc

Aprendizaje por refuerzos

Agente autónomo: percibe e interactúa con el ambiente. Tiene que elegir acciones óptimas para lograr su objetivo.

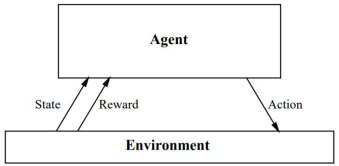
- **percibe** estado de su entorno
- realiza **acciones** para alterar ese estado
- obtiene **recompensas o castigos**



Por ej. en robot:

- **Sensores:** cámara, sonar.
- **Acciones:** adelantar, girar, cargar batería
- **Tarea:** aprender **estrategia de control** o **política** para elegir acciones que satisfagan que se carga cuando la batería está baja. Problema de **asignación de créditos**.

Aprendizaje por refuerzos



Agente realiza **acción** en **ambiente.** Obtiene un **premio** o un **castigo** en función de **cuán deseable** es el **resultado final.**

Tarea del agente:

- realizar s**ecuencias de acciones**, **observar sus consecuencias y aprender una estrategia de control**.
- Queremos aquella secuencia que **desde un estado inicial** elige **acciones que maximizan la recompensa acumulada en el tiempo**. Aprender del refuerzo (que se puede dar de manera indirecta y tardía)

Problema: sequential decision making. Se tienen que tomar decisiones todo el tiempo. Pueden tener consecuencias a largo plazo

Proceso de Decisión de Markov (MDP)

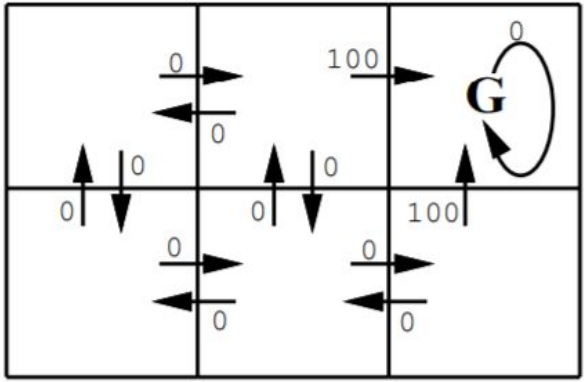
Los problemas de RL modelan el mundo usando **Markov Decision Processing (MDP)**.

MDP: <S, A, δ, R, γ>

- **S:** Conjunto de estados
- **A:** Conjunto de acciones
- **δ: S x A** $\rightarrow$ **S.** Función de transición determinística (podría ser no determinística (**probabilística**).*
- **R: S x A** $\rightarrow \mathbb{R}$ Función de recompensa
- **γ** $\in$ [0,1): factor de descuento

*Ejemplo: Robot que se mueve. Probabilidad de Norte 80%.

Ejemplo



Estados, transiciones y recompensas inmediatas

Referencias:

- **cuadrados:** estados (6).
- **flechas:** acciones, transiciones entre estados (no definido entre todos los estados) (5 opciones)
- **números:** recompensas inmediatas al hacer transición

G: estado objetivo (único en el que recibe reward). (absorbente)

Proceso de Decisión de Markov (MDP)

En MDP la **recompensa** y la **transición de un estado a otro** dependen sólo del estado actual (no de los anteriores).

- El agente percibe el **estado s_t** , elige y **realiza** una **acción a_t**.
- El **ambiente responde** dando una recompensa **$r_t = R(s_t, a_t)$.**
- $R(s_t, a_t)$ depende sólo del estado actual no de los anteriores.

Tarea: aprender política para elegir acción a_t a partir de s_t.
Se busca la **política** que arroja **mayor recompensa acumulada**.

Proceso de Decisión de Markov

- Dada una **política** o **estrategia de control: π: S -> A** (Dado un estado s en S devuelve una acción a en A)
- Se define una **función de valor:**

$$V^{\pi}(s) = r_0 + \gamma\, r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots = \sum_{t=0}^{\infty} \gamma^t r_t$$

recompensa acumulada (con descuentos) al seguir una política **π** para seleccionar acciones a partir del estado s

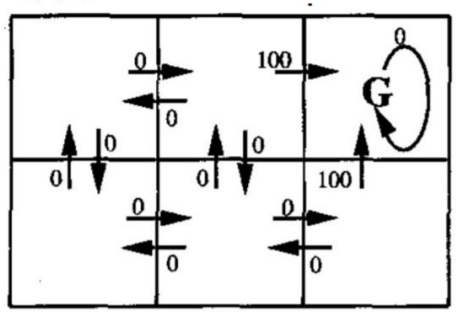
ɣ: valor de recompensas retrasadas vs. inmediatas. Ej: monetario

En esta fórmula **más peso a las inmediatas (existen otras).**
ɣ≅0 (sólo recompensa inmediata). ɣ≅1 (se da más peso a recompensas futuras)

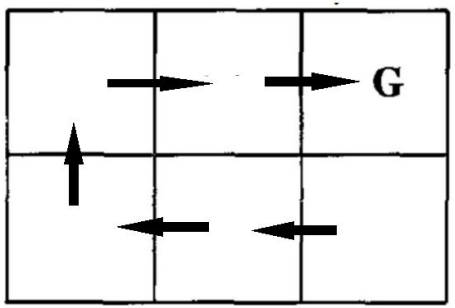
Función objetivo: política de control que maximiza la función de valor.

Ejemplo Función de Valor para política dada

R(S,A)



Política π:



Función de valor V^{π} (γ=0.9):

90	100	0
81	72.9	65.61

Función de valor, definición recursiva, función Q

¿Cómo aprender la política óptima π* para un ambiente arbitrario?

Función de valor: (1) $$V^{\pi}(s) = r_0 + \gamma\, r_1 + \gamma^2 r_2 + \ldots = \sum_{t=0}^{\infty} \gamma^t r_t$$

Definición recursiva (ecuación de Bellman) (2) $$V^{\pi}(s) = R(s, \pi(s)) + \gamma\, V^{\pi}(\delta(s, \pi(s)))$$

Objetivo: aprender política óptima **π*** (3) $$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$

(4) $$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left[R(s, a) + \gamma\, V^*(\delta(s, a)) \right]$$

Definimos (5) $$Q(s, a) \equiv R(s, a) + \gamma V^*(\delta(s, a))$$

máxima ganancia esperada desde s ejecutando a

Re-escribimos 4 (6) $$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Teníamos:

(4) $$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left[R(s,a) + \gamma V^*(\delta(s,a)) \right]$$

obtenemos (7) $$V^*(s) = \max_a \left[R(s,a) + \gamma V^*(\delta'(s,a)) \right]$$

teníamos (5) $$Q(s,a) \equiv R(s,a) + \gamma V^*(\delta(s,a))$$

podemos reescribir y obtener (8)

$$\begin{aligned} Q(s_t, a_t) &= R(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Llamamos r a R(s,a) y s' a δ (s,a) o bien s_{t+1} a δ (s_t,a)

Estrategias

- **Dilema exploración - explotación**
  - **Estrategia ε-first**
    - con probabilidad 1-ε se elige al azar (exploración)
    - con probabilidad ε se elige mejor acción conocida (explotación)
  - **Estrategia ε-greedy**
    - con probabilidad ε se elige al azar
    - con probabilidad 1-ε se elige mejor acción conocida

Azar: distribución uniforme

Q learning

Aprender la política óptima.

Inicializar tabla Q^ (s,a) con valores cero, par todos s,a.
Repetir (para cada episodio):
    s ← estado inicial
    Repetir hasta que s sea terminal:
        Elegir una acción a desde s **con una estrategia** y ejecutarla
        recompensa: r, estado nuevo: s'
        actualizar Q^ (s,a), de forma tal que
            $Q^{\wedge}(s,a) \leftarrow Q^{\wedge}(s,a) + \alpha\,(r + \gamma \max_{a'} Q^{\wedge}(s',a') - Q^{\wedge}(s,a))$
        s ← s'

α: tasa de aprendizaje α ε (0,1].

$\pi(s) = \text{argmax}_a\, Q^{\wedge}(s,a)$

estrategia: ε-first o ε-greedy

Deep reinforcement learning

- Aproximación de Q(s,a) con una red neuronal convolucional (CNN).

Resumen

Características de aprendizaje por refuerzos:

- **recompensa tardía:** queremos aprender π, tq dado un estado s, a = π(s). No tenemos como entrada el par (s, π(s))
- **estrategias de experimentación:** dilema **exploración** (nuevos) **vs. explotación** (conocidos). el “entrenamiento” está dado por la secuencia de acciones que se sigue.
- estados parcial o totalmente observables
- aprendizaje permanente

Resumen

- Interacción con ambiente.
- **Aprendizaje con un crítico** en vez de aprendizaje con un **maestro**. No avisa de antemano qué hacer. Poco feedback y tardío.
- Proceso de decisión de Markov (MDP): MDP: <S, A, δ, R, γ>
- Política π, Función de valor Vπ(s), Función Q(s,a)
- Estrategias ε-greedy y ε-first
- Algoritmo Q-Learning

Bibliografía

Capítulos de libros:

Mitchell, cap. 13

Alpaydin, cap. 18

Libros enteros:

Reinforcement Learning. An Introduction. Sutton, Barto.

Algorithms for reinforcement learning. Szepesváry

Cursos:

UCL: Reinforcement learning. David Silver:
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

CS229 Stanford Andrew NG: https://www.youtube.com/watch?v=RtxI449ZjSc