Aprendizaje por refuerzo

Sistemas Inteligentes Distribuidos

Sergio Alvarez
Javier Vázquez

Bibliografía

- *Artificial intelligence: a modern approach* (Russell & Norvig), cap. 23
- *Reinforcement Learning: An Introduction* (Sutton & Barto), cap. 2, 6
- *Multi-Agent Reinforcement Learning* (Albrecht et al.), cap. 2, 8

Máquinas tragaperras

Aprendizaje por refuerzo

Repasando lo visto hasta ahora

- Para cada turno $t = 1, 2, \ldots$
  - Escoge: **jugar** o **parar**
  - Efecto de **jugar:**
    - Recibes 4 euros
    - Lanzas un dado de 6 caras
      - Si el resultado es 1 o 2, se acaba el juego
      - Si el resultado es otro, se avanza al siguiente turno
  - Efecto de **parar:**
    - Recibes 10 euros
    - Se acaba el juego

Sabemos cómo resolver este tipo de tarea

¿Cómo?

¿Por qué?

Vamos a jugar a otro juego

- Utilidad $\mathcal{U}_0 = 0$
- Para cada turno $t = 1, 2, \ldots, 20$
  - Escoge: **jugar1, jugar2** o **parar**
  - Efecto de **parar**:
    - Se acaba el juego con utilidad $\mathcal{U}_{t-1}$
  - Efecto de **jugarX**:
    - Se suma r_t^X (positiva o negativa) a $\mathcal{U}_t$

Multi-armed bandits

- ¿Cómo podemos abordar este tipo de tarea?
  - Intuición: **aprender** la ganancia esperada con cada decisión (estado-acción)
- El dilema: **exploración vs explotación**

Aprender a decidir

Aprendizaje por refuerzo

Contexto

- Supongamos las siguientes premisas:
  - Tenemos una representación del estado
  - Tenemos el conjunto de acciones disponibles
  - No tenemos acceso a:
    - La función de recompensa, o
    - La función de transición entre estados a través de pares estado-acción
- Básicamente, no tenemos un MDP completo
  - El agente está *dentro* del MDP, pero no tiene total acceso a él
- El **aprendizaje por refuerzo** es el área que se encarga de solucionar problemas formulados de esta manera

Tipos de aprendizaje por refuerzo

- **Aprendizaje activo**
  - El agente **aprende una política**
  - La tarea del agente está en equilibrar exploración vs explotación
- **Aprendizaje pasivo**
  - La política está fijada
  - La tarea del agente es **aprender la utilidad** de los estados o los pares estado-acción
- **Aprendizaje online**
  - El agente interactúa con el entorno, y al mismo tiempo, aprende **a través de esta interacción**
- **Aprendizaje offline**
  - El agente aprende **a partir de una colección acumulada de experiencias** sobre el entorno

Tipos de aprendizaje por refuerzo

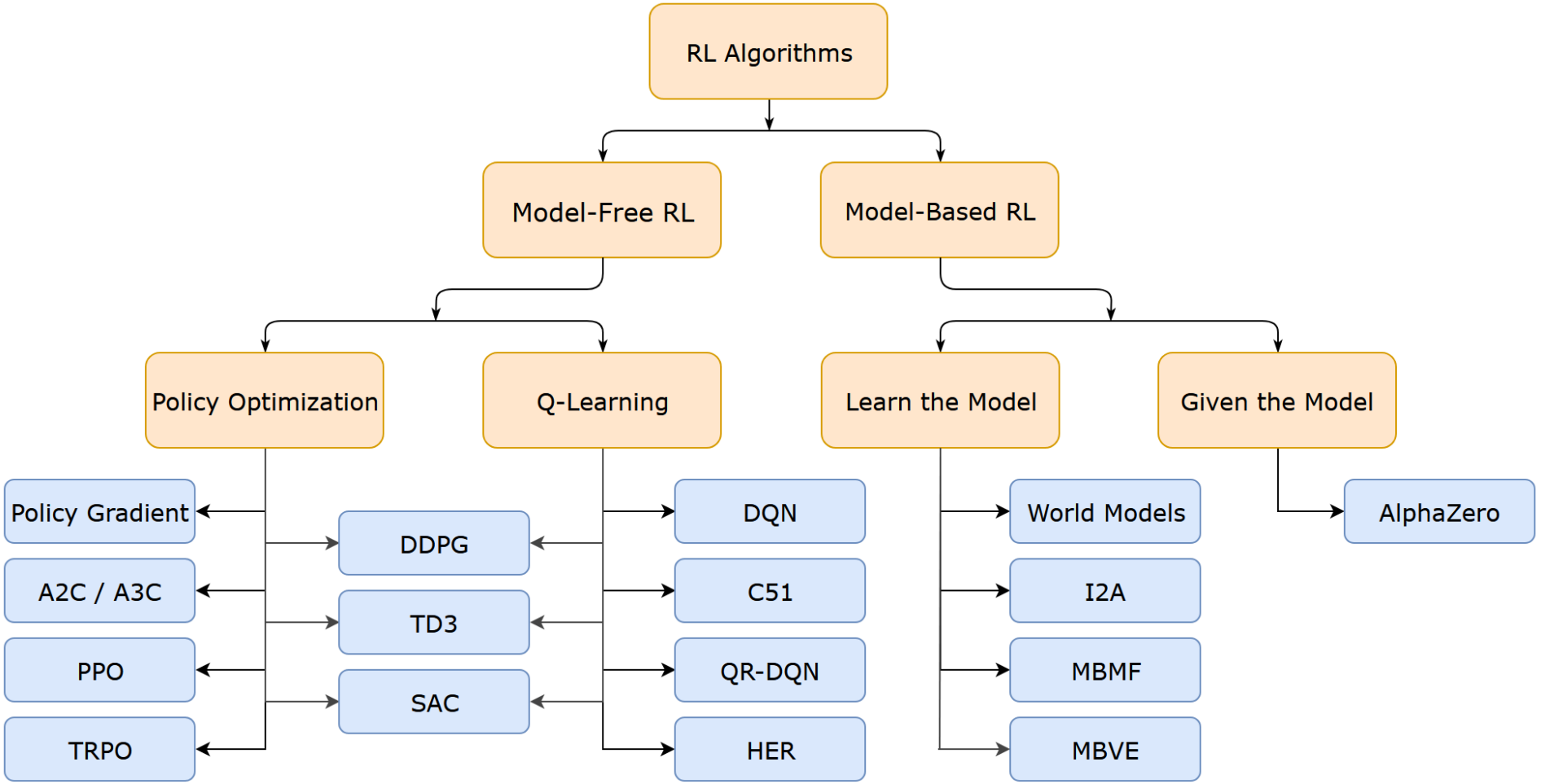
- **Aprendizaje *on-policy***
  - La política se aprende y se mejora **a partir de los mismos datos y acciones que la propia política genera**
- **Aprendizaje *off-policy***
  - La política se aprende **al margen de la que se usa para interactuar** con el entorno

- **Aprendizaje basado en modelo** (*model-based*)
  - El agente **usa o construye un modelo**, generalmente un MDP, sobre el cual se aprende
- **Aprendizaje no basado en modelo** (*model-free*)
  - El agente aprende una política sin un modelo y por lo tanto **sin realizar estimaciones de transiciones o recompensas**

Tipos de aprendizaje por refuerzo

- **Aprendizaje basado en valor (*value-based*)**
  - La política se aprende como resultado derivado de un proceso en el que se calcula y evoluciona una función de valor, ya sea valor de estado (V o V^*) o valor de estado-acción (Q o Q^*)
- **Aprendizaje basado en política (*policy-based*)**
  - La política se aprende de manera directa, sin el cálculo intermedio de una función de valor
- **Aprendizaje híbrido valor-política**
  - En el proceso de aprendizaje de la política, se combinan los dos enfoques anteriores

Una taxonomía



https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Una taxonomía

- Vamos a ver cuatro métodos paradigmáticos que son la base de la mayoría de métodos vigentes:
  - Basado en modelo
    - **Model-based Monte Carlo**: offline, on-policy, value-based
  - No basados en modelo
    - **SARSA**: online, on-policy, value-based
    - **Q-learning**: online, off-policy, value-based
    - **REINFORCE**: online/offline, on-policy, policy-based
- Estos cuatro métodos son de aprendizaje activo
  - Métodos de aprendizaje pasivo: **evaluación de política** (*aprendemos* el valor de los estados), evaluación Monte Carlo, evaluación directa…

Model-based Monte Carlo

Aprendizaje por refuerzo

Aprendizaje basado en modelo

- Si ya tenemos un modelo: **iteración de valor** (*value iteration*)
  - Se considera un método de aprendizaje por refuerzo ya que se **aprende una política**
  - Alternativas: **iteración de política** (*policy iteration*), **búsqueda de política** (*policy search*)
- Si no tenemos un modelo completo (nuestras premisas):
  - Monte Carlo basado en modelo (*Model-based Monte Carlo*)
  - *World Models* (Ha and Schmidhuber, 2018): combinación de aprendizaje supervisado para percepción con un modelo entrenado de estados futuros y recompensas
  - *Imagination-Augmented Agents* (*I2A*, Weber et al, 2017): aprendizaje de la dinámica del entorno para posteriormente “imaginar” resultados para posibles secuencias de acciones
  - …

Monte Carlo (basado en modelo)

- La idea es simular trayectorias del agente en el entorno y obtener:

$$s_0, a_0, r_0; s_1, a_1, r_1; s_2, a_2, r_2; \ldots; s_n, a_n, r_n$$

- Y a partir de un conjunto de estas trayectorias estimar $\mathcal{T}$ y $\mathcal{R}$:

$$\hat{\mathcal{T}}(s, a, s') = \frac{|\{t \mid s_t = s, a_t = a, s_{t+1} = s'\}|}{|\{t \mid s_t = s, a_t = a\}|}$$

$$\hat{\mathcal{R}}(s, a, s') = r \in \{r_t | s_t = s, a_t = a, s_{t+1} = s'\}$$

Monte Carlo (basado en modelo)

- La exploración en este caso se vuelve **necesaria**
  - O habrá estados que nunca se visiten y por lo tanto no formen parte de transiciones o recompensas
- El algoritmo de búsqueda Monte Carlo define una estrategia para decidir cómo se realiza la simulación y por lo tanto determinar qué trayectorias se extraen
- Una vez tenemos una estimación de $\mathcal{T}$ y $\mathcal{R}$, podemos usar los algoritmos que ya conocemos para obtener la política óptima
- Hay una versión *model-free* de este método

TD, SARSA, Q-Learning

Aprendizaje por refuerzo

Aprendizaje por diferencias temporales

- El aprendizaje por diferencias temporales (*Temporal Difference* o *TD Learning*) es la base de muchos métodos como SARSA o Q-learning
- Idea principal: aprender de la experiencia a partir de π
  - Generar muestras de estados, acciones y recompensas: $s, \pi(s), s', r$

$$\mathcal{X} = \mathcal{R}(s, \pi(s), s') + \gamma V^{\pi}(s')$$

  - Y actualizar $V^{\pi}(s)$ de manera constante, aplicando una media móvil:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(\mathcal{X} - V^{\pi}(s))$$

donde $\alpha \in (0,1]$ es la tasa de aprendizaje o *learning rate*

Aprendizaje por diferencias temporales

- La forma general de la familia de algoritmos TD se basa en este tipo de formulas de actualización, aunque más adelante veremos versiones para actualizar Q directamente
- La variable $\mathcal{X}$ es comúnmente llamada *muestra* u *objetivo de actualización* (*update target)* y la forma que hemos visto identifica al algoritmo como **TD(0)** ya que únicamente usa la recompensa inmediata y la estimación actual de $V(s')$:

$$\mathcal{X} = \mathcal{R}(s, \pi(s), s') + \gamma V^{\pi}(s')$$

- Observad que **no necesitamos conocimiento previo de la función de transición ni de la función completa de recompensa**

SARSA

- El algoritmo SARSA (Sutton and Barto, 2018) incorpora la ecuación Q de Bellman para generar un nuevo algoritmo TD. La forma general de esta ecuación es:

$$Q^{\pi}(s,a) = \sum_{s' \in S} p(s'|s,a) \left[\mathcal{R}(s,a,s') + \gamma \overbrace{\sum_{a' \in A} \pi(a'|s) Q^{\pi}(s',a')}^{V^{\pi}(s')} \right]$$

- En el caso de SARSA, generamos trayectorias para capturar tuplas:

$$(s, a, r, s', a')$$

SARSA

- La fórmula de actualización para Q^{π} es:

$$Q^{\pi}(s,a) \leftarrow Q^{\pi}(s,a) + \alpha\big(\mathcal{X} - Q^{\pi}(s,a)\big)$$

- Basándose en la ecuación de Bellman y la información recogida, la regla de actualización queda de la siguiente manera:

$$\mathcal{X} = \mathcal{R}(s,a,s') + \gamma Q^{\pi}(s',a')$$

- Con lo que la actualización tras cada nueva muestra es:

$$Q^{\pi}(s,a) \leftarrow Q^{\pi}(s,a) + \alpha(\mathcal{R}(s,a,s') + \gamma Q^{\pi}(s',a') - Q^{\pi}(s,a))$$

Aproximando el algoritmo SARSA

- Inicialización
  - Inputs: tasa de aprendizaje $\alpha \in (0,1]$
  - $Q(s,a) = 0 \ \ \forall s \in S, a \in A$
- Para cada episodio:
  - Observar estado s_0
  - Para cada $t = 0, 1, 2, \dots, t_{MAX}$
    - Ejecutar acción $a_t \in \arg\max_{a \in A} Q(s_t, a)$
    - Observar estado s_{t+1} y recompensa r_t
    - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\mathcal{R}(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

Aproximando el algoritmo SARSA

- Inicialización
  - Inputs: tasa de aprendizaje $\alpha \in (0,1]$
  - $Q(s,a) = 0 \ \ \forall s \in S, a \in A$
- Para cada episodio:
  - Observar estado s_0
  - Para cada $t = 0, 1, 2, \dots, t_{MAX}$
    - Ejecutar acción $a_t \in \arg\max_{a \in A} Q(s_t, a)$
    - Observar estado s_{t+1} y recompensa r_t
    - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\mathcal{R}(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

¿Qué problema puede tener este algoritmo?

Algoritmo SARSA (ϵ-greedy)

- Inicialización
  - Inputs: tasa de aprendizaje $\alpha \in (0,1]$, coeficiente de **exploración** $\epsilon > 0$
  - $Q(s,a) = 0 \ \forall s \in S, a \in A$
- Para cada episodio:
  - Observar estado s_0
  - Para cada $t = 0, 1, 2, \dots, t_{MAX}$
    - Con probabilidad ϵ:
      - Ejecutar acción aleatoria $a_t \in A$
      - Si no, ejecutar acción $a_t \in \arg\max_{a \in A} Q(s_t, a)$
    - Observar estado s_{t+1} y recompensa r_t
    - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\mathcal{R}(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

Algoritmo SARSA (ϵ-greedy)

- La política según este algoritmo se define como:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|A|} & \text{si } a \in \arg\max_{a' \in A} Q(s, a') \\ \dfrac{\epsilon}{|A|} & \text{en cualquier otro caso} \end{cases}$$

Algoritmo SARSA: optimalidad

- Hay garantía de que se aprende π^* si se cumplen:
  - Todas las combinaciones $(s, a) \in S \times A$ se prueban un número infinito de veces
    - e.g. $\epsilon > 0$ pero converge a $\epsilon = 0$ en el infinito
  - La tasa de aprendizaje cumple las siguientes propiedades:

$$\forall s \in S, a \in A: \sum_{k=0}^{\infty} \alpha_k(s,a) \to \infty \ \wedge \sum_{k=0}^{\infty} \alpha_k(s,a)^2 < \infty$$

- La tasa de aprendizaje puede variar en el tiempo, buscando aproximaciones menos precisas y más rápidas al inicio, y más estables al final:
  - $\alpha_k(s,a) = c$, donde c es una constante, **no** es un buen *learning rate*
  - $\alpha_k(s,a) = \frac{1}{k}$, en cambio, sí es un buen *learning rate*

Q-learning

- Q-learning (Watkins and Dayan, 1992) también define una actualización, pero esta vez sobre la ecuación Q^*:

$$Q^*(s,a) = \sum_{s' \in S} p(s'|s,a) \left[\mathcal{R}(s,a,s') + \gamma \underbrace{\max_{a' \in A} Q^*(s',a')}_{V^*(s')} \right]$$

- En este caso, únicamente es necesaria la tupla (s, a, r, s')
- La regla de actualización correspondiente es:

$$\mathcal{X} = \mathcal{R}(s,a,s') + \gamma \max_{a \in A} Q(s',a)$$

Algoritmo Q-learning (ϵ-greedy)

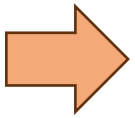
- Inicialización
  - Inputs: tasa de aprendizaje $\alpha \in (0,1]$, coeficiente de exploración $\epsilon > 0$
  - $Q(s,a) = 0 \ \forall s \in S, a \in A$
- Para cada episodio:
  - Observar estado s_0
  - Para cada $t = 0, 1, 2, \dots, t_{MAX}$
    - Con probabilidad ϵ:
      - Ejecutar acción aleatoria $a_t \in A$
      - Si no, ejecutar acción $a_t \in \arg\max_{a \in A} Q(s_t, a)$
    - Observar estado s_{t+1} y recompensa r_t
    - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[\mathcal{R}(s_t, a_t, s_{t+1}) + \gamma \max_{a'} \boldsymbol{Q(s_{t+1}, a')} - Q(s_t, a_t) \right]$

Q-learning es un método *off-policy*

- Al estar siguiendo la ecuación para Q^*,
  - y por lo tanto aprender $Q(s_t, a_t)$ a partir de $\boldsymbol{\max_{a'} Q(s_t, a')}$
  - en lugar de aprender $Q(s_t, a_t)$ a partir de $\boldsymbol{Q(s_{t+1}, a_{t+1})}$, como en SARSA
- **…no necesitamos utilizar la función Q (que estamos aprendiendo) como base de la política del algoritmo**

Con probabilidad ϵ:

- Escoger acción aleatoria $a_t \in A$
- Si no, ejecutar acción $a_t \in \arg\max_{a \in A} Q(s_t, a)$



Ejecutar acción $a_t \sim \pi(\cdot \mid s_t)$

Algoritmo Q-learning (ϵ-greedy)

- Inicialización
  - Inputs: tasa de aprendizaje $\alpha \in (0,1]$, coeficiente de exploración $\epsilon > 0$, **política $\boldsymbol{\pi}$**
  - $Q(s,a) = 0 \ \ \forall s \in S, a \in A$
- Para cada episodio:
  - Observar estado s_0
  - Para cada $t = 0, 1, 2, \dots, t_{MAX}$
    - Ejecutar acción $a_t \in a_t \sim \pi(\cdot \mid s_t)$
    - Observar estado s_{t+1} y recompensa r_t
    - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[\mathcal{R}(s_t, a_t, s_{t+1}) + \gamma \max_{\boldsymbol{a'}} \boldsymbol{Q}(\boldsymbol{s_{t+1}}, \boldsymbol{a'}) - Q(s_t, a_t)\right]$

Algoritmo Q-learning: optimalidad

- La política aprendida se define como:

$$\pi(a|s) = \begin{cases} 1 & \text{si } a = \arg\max_{a' \in A} Q(s, a') \\ 0 & \text{en cualquier otro caso} \end{cases}$$

- Hay garantía de que se aprende π^* si se cumplen:
  - Todas las combinaciones $(s, a) \in S \times A$ se prueban un número infinito de veces
    - e.g. $\epsilon > 0$ pero converge a $\epsilon = 0$ en el infinito
  - La tasa de aprendizaje cumple las siguientes propiedades:

$$\forall s \in S, a \in A: \sum_{k=0}^{\infty} \alpha_k(s, a) \to \infty \ \wedge \sum_{k=0}^{\infty} \alpha_k(s, a)^2 < \infty$$

Métodos TD

- Los algoritmos basados en el aprendizaje por diferencias temporales son fundamentales en el aprendizaje por refuerzo
- TD introduce la idea de aprender directamente de la experiencia sin necesidad de un modelo del entorno
- Estos algoritmos permiten flexibilidad en el equilibrio entre exploración y explotación, principalmente mediante ϵ

REINFORCE

Aprendizaje por refuerzo

Optimizar una política

- Nuestro objetivo es maximizar la utilidad esperada para una trayectoria $[s_0, a_0, s_1, a_1, a_2, \dots]$ a partir de una política $\pi(a|s)$:

$$\mathbb{E}_\pi[\mathcal{U}([s_0, a_0, s_1, a_1, a_2, \dots])] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

- Podemos definir una función J con una parametrización θ tal que:

$$J(\theta) = \mathbb{E}_\pi[\mathcal{U}([s_0, a_0, s_1, a_1, a_2, \dots])]$$

- A la política resultante de aplicar una θ la denominaremos $\pi(a|s;\theta)$
  - Del mismo modo que a partir de $V^*(s)$ podíamos obtener $\pi^*(a|s)$

Gradiente de política

- El problema de optimización de una política se puede ver pues como la búsqueda de una θ que maximice $J(\theta)$
- Esta función J puede ser, por ejemplo, un sistema de ecuaciones lineales o una red neuronal
- Para resolver este problema podemos utilizar métodos clásicos del aprendizaje automático, como por ejemplo el ascenso de gradiente:

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

donde α es la tasa de aprendizaje o *learning rate*, y $\nabla J(\theta_t)$ representa el gradiente de la función J con respecto de la parametrización θ_t

Teorema del gradiente de política

- El *teorema de gradiente de política* (Sutton and Barto, 2018) encuentra una formulación de este gradiente tal que:

$$\nabla_\theta J(\theta) \propto \sum_{s \in S} \Pr(s|\pi) \sum_{a \in A} Q^\pi(s,a) \nabla_\theta \pi(a|s;\theta)$$

donde

$a \propto b$ indica que la variable a es proporcional a la variable b,

$\Pr(s|\pi)$ es la distribución de probabilidad de que un estado s sea visitado por la política π (comenzando la trayectoria por un estado inicial que sigue μ)

$\nabla_\theta \pi(a|s;\theta)$ es el gradiente de la política, apuntando en la dirección del espacio de parámetros que incrementa la probabilidad de escoger a en s

REINFORCE

- El algoritmo REINFORCE (Williams, 1992) utiliza una función de pérdida para implementar este gradiente de manera eficiente que tiene en cuenta la trayectoria $[s_0, a_0, r_0, \dots, s_T, a_T, r_T]$ de un episodio:

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=0}^{T-1}\left(\sum_{t'=t}^{T-1}\gamma^{t'-t}\,\mathcal{R}(s_t, a_t, s_{t+1})\right)\log\pi(a_t|s_t;\theta)$$

- Observad que no necesitamos:
  - Función de transición entre estados y pares estado-acción
  - Cálculos intermedios de las funciones de valor V o Q

El algoritmo REINFORCE

- Inicialización:
  - Función J con parámetros iniciales θ
  - Política π, derivada de $J(\theta)$
- Para cada episodio:
  - Para cada $t = 1, 2, \ldots, T-1$:
    - Observar estado s_t
    - Ejecutar acción $a_t \sim \pi(\cdot \mid s_t; \theta)$
    - Observar recompensa r_t, estado s_{t+1}
  - Actualizar parámetros θ, **maximizando** $\mathcal{L}(\theta)$
    - $\theta \leftarrow \theta + \alpha \cdot \nabla J(\theta_t)$

El algoritmo REINFORCE

- Es un algoritmo simple que ha servido como fundamento para numerosas propuestas en el contexto del aprendizaje por gradiente de política
- En REINFORCE, se suele utilizar el método de simulación de Monte Carlo para generar las trayectorias
- Por su metodología *end-to-end* (a partir de trayectorias completas hasta un tiempo T), puede adaptarse para *offline*
- Sin embargo, es vulnerable frente a la alta varianza que puede surgir entre las utilidades generadas por episodios diferentes

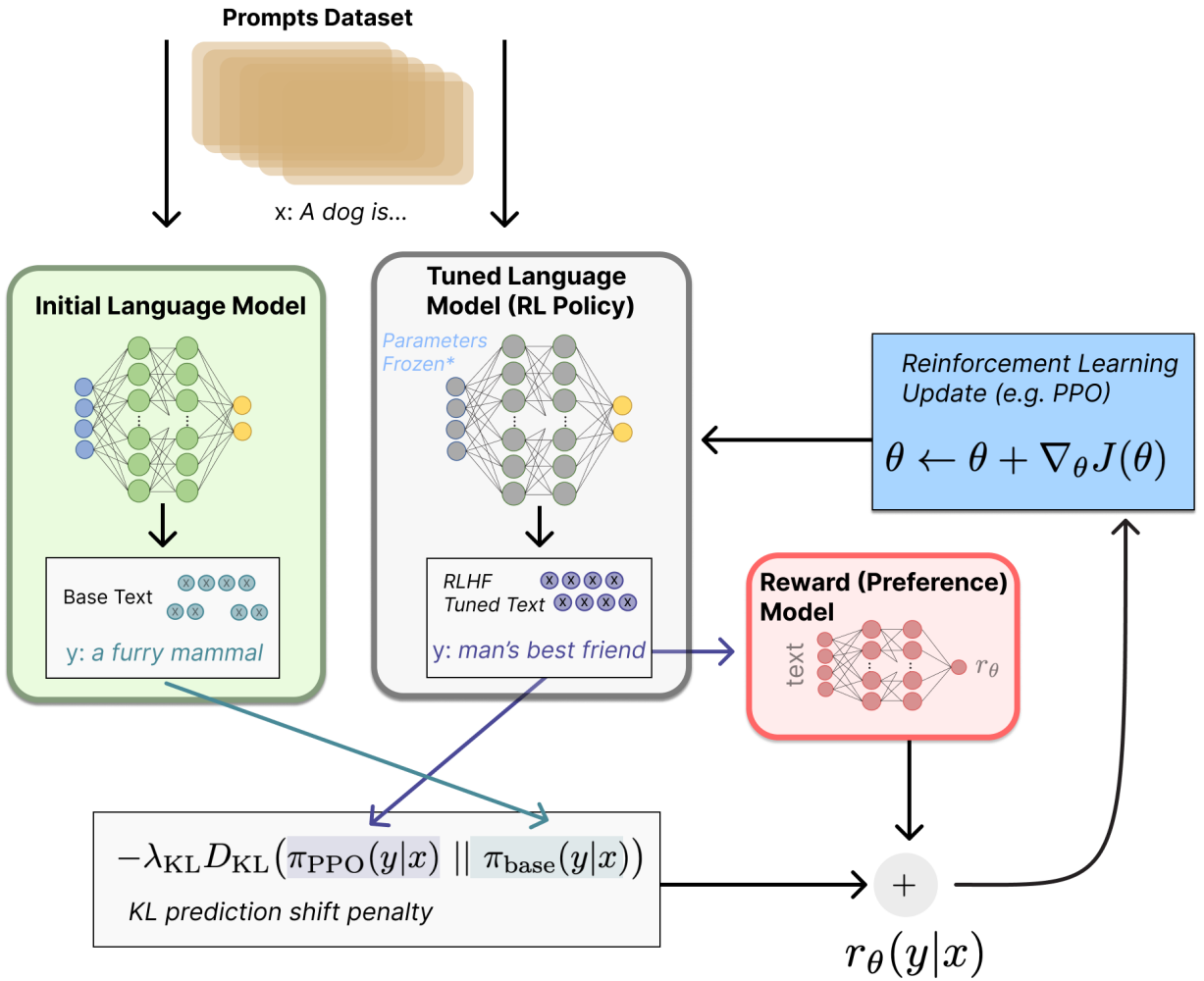
Estado del arte

Aprendizaje por refuerzo

Algunos algoritmos

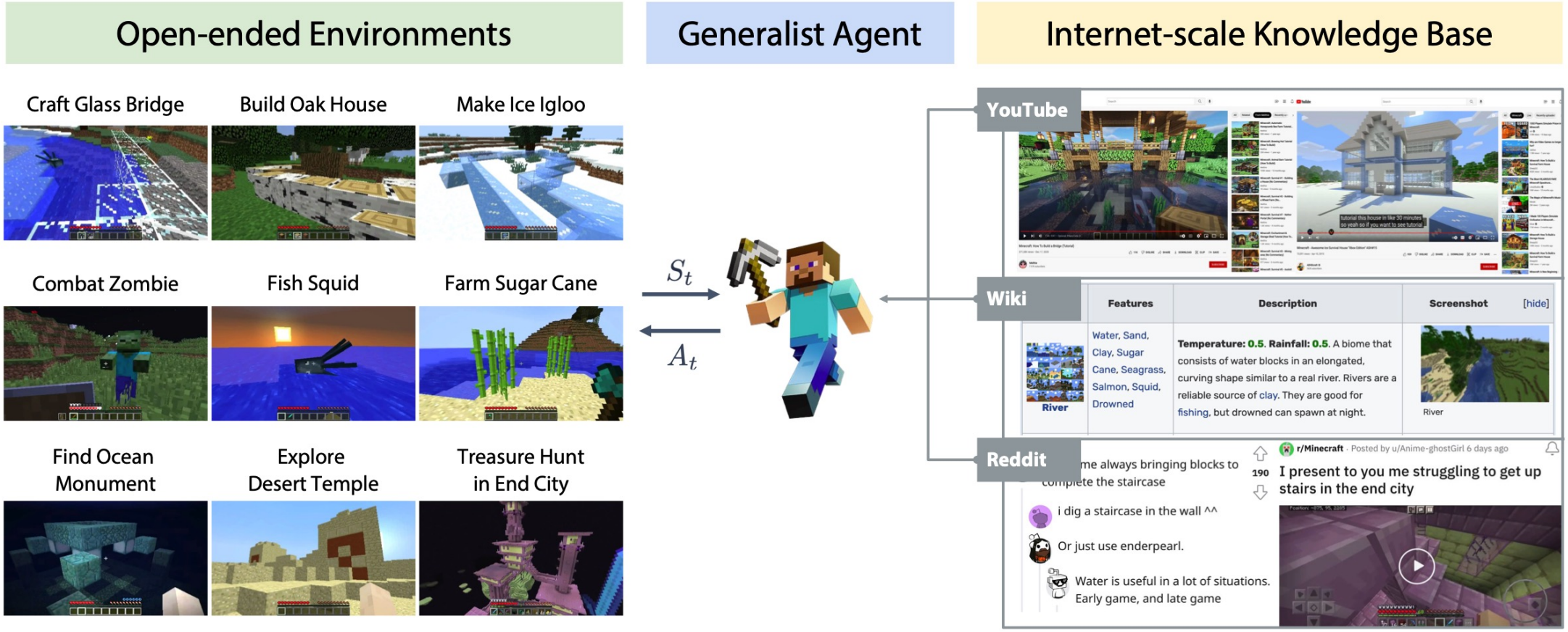
- **Deep Q-Network (DQN; Mnih et al, 2013):** extensión de Q-learning con redes neuronales para aproximar la función Q a partir de un buffer de experiencia (*replay buffer*)
- **Proximal Policy Optimization (PPO; Schulman et al, 2017):** evolución de ciertos algoritmos de gradiente de política como REINFORCE, con ciertas optimizaciones que permiten una actualización de política más estable, fiable y eficiente
- **Algoritmos Actor-Critic (A3C, A2C, SAC, DDPG):** híbridos en el sentido de que combinan un método *policy-based* que propone acciones (*actor*) y un método *value-based* que evalúa las acciones (*critic*) que se retroalimentan
- **AlphaZero (Silver et al, 2017):** algoritmo auto-supervisado basado en modelo (búsqueda Monte Carlo) que no necesita datos históricos, combinando la búsqueda Monte Carlo con representaciones intermedias usando redes neuronales profundas que modelan valor y política

Reinforcement Learning with Human Feedback



https://huggingface.co/blog/rlhf

MineDojo



Fan et al., *MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge*, https://arxiv.org/abs/2206.08853

Alternativas

- Aprendizaje por imitación (*Imitation Learning*)
  - Cuando no existe una señal de recompensa, pero sí existen ejemplos de comportamiento racional
    - Aprendizaje por demostración o por replicación
    - Aprendizaje inverso
- Aprendizaje por currículum (*Curriculum Learning*)
  - El aprendizaje se produce usando tareas o habilidades como elementos de primer orden, empezando por tareas o habilidades sencillas (posiblemente prediseñadas) y creando nuevas gradualmente más complejas (e.g. *skills*)