Algoritmos RL

Alumno: Cristhian Wiki Sánchez Sauñe

(a) Algoritmo GPI (Generalized Policy Iteration)

Este algoritmo hace referencia a un procedimiento iterativo para mejorar la política al combinar la evaluación y mejora de políticas anteriores.

$$\pi_0 \xrightarrow{\text{evaluacion}} V_{\pi_0} \xrightarrow{\text{mejora}} \pi_1 \xrightarrow{\text{evaluacion}} \dots \xrightarrow{\text{mejora}} \pi_* \xrightarrow{\text{evaluacion}} V_*$$

En el algoritmo GPI, la función value se aproxima repetidamente para estar más cerca del valor real de la política actual y, mientras tanto, la política se mejora repetidamente para abordar la optimización. Este proceso de iteración de políticas funciona y siempre converge con la optimización.

Para conocer cómo surge la convergencia anterior, digamos que tenemos una política **π** y luego generamos una versión mejorada **π ′** tomando acciones greedy:

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q_\pi(s, a)$$

Se garantiza que el valor de este π ′ mejorado será mejor porque:

$$\begin{aligned} Q_\pi(s, \pi'(s)) &= Q_\pi(s, \arg\max_{a \in \mathcal{A}} Q_\pi(s, a)) \\ &= \max_{a \in \mathcal{A}} Q_\pi(s, a) \geq Q_\pi(s, \pi(s)) = V_\pi(s) \end{aligned}$$

(b) Método Monte Carlo (MC). Explica como se usa el algoritmo GPI con los métodos de Montecarlo.

Recordando la siguiente fórmula: $V(s) = \mathbb{E}[G_t | S_t = s]$

Los métodos de Monte-Carlo (MC) utilizan una idea simple: aprende de episodios de experiencia bruta sin modelar la dinámica ambiental y calcula el rendimiento medio observado como una aproximación del rendimiento esperado. Para calcular el rendimiento empírico **G***t*, los métodos de MC deben aprender de $S_1, A_1, R_2, \dots, S_T$ episodios completos para poder calcular:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

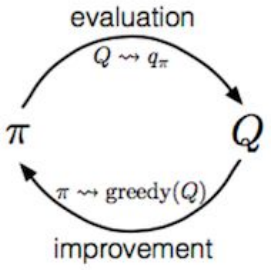
y todos los episodios deben eventualmente terminar.
El rendimiento medio empírico para el estado **s** es:

$$V(s) = \frac{\sum_{t=1}^{T} \mathbb{1}[S_t = s] G_t}{\sum_{t=1}^{T} \mathbb{1}[S_t = s]}$$

donde $\mathbb{1}[S_t = s]$ es una función de indicador binario. Podemos contar la visita de los estados cada vez para que puedan existir múltiples visitas de un estado en un episodio ("cada visita"), o solo contarla la primera vez que encontramos un estado en un episodio ("primera visita "). Esta forma de aproximación se puede extender fácilmente a funciones de valor de acción contando un par **(s, a)**.

$$Q(s, a) = \frac{\sum_{t=1}^{T} \mathbb{1}[S_t = s, A_t = a] G_t}{\sum_{t=1}^{T} \mathbb{1}[S_t = s, A_t = a]}$$

Para conocer la política óptima de MC, la iteramos siguiendo una idea similar a **GPI** (mencionada anteriormente, de allí su relación).



1. Mejorar la política de forma greedy con respecto a la función de valor actual:

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$$

2. Generar un nuevo episodio con la nueva política **π** (es decir, usar algoritmos como **ε-greedy** nos ayuda a equilibrar la explotación y la exploración).
3. Estimar Q usando el nuevo episodio:

$$q_\pi(s, a) = \frac{\sum_{t=1}^{T} \left(\mathbb{1}[S_t = s, A_t = a] \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \right)}{\sum_{t=1}^{T} \mathbb{1}[S_t = s, A_t = a]}$$

(c) Algoritmo SARSA

"SARSA" se refiere al procedimiento de actualizar el Q-value siguiendo una secuencia de $\ldots, S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \ldots$

La idea sigue la misma ruta de **GPI** (mencionada anteriormente). Dentro de un episodio, funciona de la siguiente manera:

1. Inicializamos **t=0**
2. Comenzamos con S_0 y tomamos una acción $A_0 = \arg\max_{a \in \mathcal{A}} Q(S_0, a)$
3. En el momento **t**, después de aplicar la acción **A**t, observamos la recompensa R_{t+1} y pasamos al siguiente estado S_{t+1}
4. Luego, elegimos la siguiente acción de la misma manera que en el paso 2:
$$A_{t+1} = \arg\max_{a \in \mathcal{A}} Q(S_{t+1}, a)$$
5. Actualizamos la función Q-value:
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$
6. Establecemos t = t + 1 y repetimos desde el paso 3.

En cada paso de SARSA, debemos elegir la siguiente acción de acuerdo con la política actual.

(d) Algoritmo de Actor-Critic

Si se aprende la función de valor además de la política, obtendremos el algoritmo Actor-Critic.

- **Crítico**: actualiza los parámetros **w** de la función de valor, dependiendo del algoritmo, podría ser el valor de acción **Q(a | s; w)** o el valor de estado **V(s; w)**.
- **Actor**: actualiza los parámetros **θ** de la política, en la dirección sugerida por el crítico, **π(a | s; θ)**.

Esta idea de actualizar los parámetros a partir de las sugerencias de otra red, también la vemos en el área del Deep Learning, más específicamente en las Redes Generativas Adversarias.

Veamos cómo funciona un algoritmo Actor-Critic acción-valor:

1. Inicialice **s**, **θ**, **w** al azar; muestramos $a \sim \pi(a|s;\theta)$
2. Para t = 1… T:
    a. Muestreamos la recompensa $r_t \sim R(s,a)$ y el siguiente estado

$$s' \sim P(s'|s,a)$$

b. Muestreamos la siguiente acción $a' \sim \pi(s', a'; \theta)$

c. Actualizamos los parámetros de la política:

$$\theta \leftarrow \theta + \alpha_\theta Q(s,a;w)\nabla_\theta \ln \pi(a|s;\theta)$$

d. Calculamos la corrección acción-valor en el tiempo **t**:

$$G_{t:t+1} = r_t + \gamma Q(s', a'; w) - Q(s, a; w)$$

y lo utilizamos para actualizar los parámetros de la función de valor:

$$w \leftarrow w + \alpha_w G_{t:t+1} \nabla_w Q(s,a;w)$$

e. Actualizamos la acción y el estado:

$$a \leftarrow a' \qquad\qquad s \leftarrow s'$$

α_θ y α_w son dos tasas de aprendizaje para las actualizaciones de parámetros de función de valor y política, respectivamente.