Introducción al Aprendizaje por Refuerzo

Este documento presenta las definiciones y conceptos clave del Aprendizaje por Refuerzo (RL, por sus siglas en inglés). Está diseñado para un curso de Inteligencia Artificial, cubriendo los fundamentos matemáticos y proporcionando intuición y ejemplos sencillos.

---

1. Agente y Entorno

- **Agente**: El que aprende o toma decisiones.
- **Entorno**: Todo con lo que el agente interactúa.

En cada paso discreto de tiempo $t = 0, 1, 2, \dots$:

1. El agente observa un **estado** S_t del entorno.
2. Elige una **acción** A_t de acuerdo a una **política** π.
3. El entorno responde con un nuevo estado S_{t+1} y una **recompensa** R_{t+1}.

Esta interacción continúa hasta que se cumpla una condición terminal o indefinidamente.

---

2. Proceso de Decisión de Markov (MDP)

Los problemas de RL suelen modelarse como un MDP, definido por la tupla $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- **Espacio de estados** $\mathcal{S}$: conjunto de todos los estados posibles.
- **Espacio de acciones** $\mathcal{A}$: conjunto de todas las acciones posibles.
- **Probabilidad de transición** $P(s'|s,a) = \Pr{S_{t+1}=s'|S_t=s, A_t=a}$.
- **Función de recompensa** $R(s,a,s') = \mathbb{E}[R_{t+1}|S_t=s, A_t=a, S_{t+1}=s']$.
- **Factor de descuento** $\gamma \in [0,1]$: pondera las recompensas futuras.

PROF

Ejemplo (Mundo de cuadrícula): Una cuadrícula de 4×4, los estados son celdas, las acciones son {arriba, abajo, izquierda, derecha}. Chocar contra la pared mantiene al agente en la misma celda. Recompensas: -1 por cada paso, +10 al llegar a la meta.

---

3. Política π

Una política define el comportamiento del agente:

- **Determinista**: $\pi : \mathcal{S} \to \mathcal{A}$, elige una acción específica para cada estado.
- **Estocástica**: $\pi(a|s) = \Pr{A_t = a \mid S_t = s}$.

Las políticas pueden estar parametrizadas por θ (por ejemplo, pesos de una red neuronal) como $\pi_\theta(a|s)$.

3.1 Políticas Parametrizadas

Una política parametrizada π_θ representa la estrategia del agente mediante un conjunto de parámetros θ que pueden ajustarse. Imagina que estamos creando un agente para jugar ajedrez: los parámetros θ podrían determinar qué tanto valora el agente cada pieza (peones, alfiles, torres) o posiciones específicas en el tablero. Si usamos una red neuronal, θ correspondería a los pesos de las conexiones entre neuronas. Lo poderoso de este enfoque es que, en lugar de programar manualmente reglas para cada situación posible, podemos optimizar estos parámetros mediante algoritmos de aprendizaje.

Por ejemplo, una forma simple de parametrizar sería asignar un valor a cada tipo de pieza: $\theta = [\theta_{\text{peón}}, \theta_{\text{caballo}}, \theta_{\text{alfil}}, \theta_{\text{torre}}, \theta_{\text{reina}}, \theta_{\text{rey}}]$ donde quizás inicialmente $\theta = [1, 3, 3, 5, 9, 100]$. La política podría elegir movimientos que maximicen la suma de piezas propias menos las del oponente, ponderadas por estos valores. El aprendizaje consistiría en ajustar estos valores basándose en los resultados de las partidas.

Ejemplo simple: Política ε-greedy para bandido

La heurística más sencilla para el problema del bandido multibrazo (elegir entre varias máquinas tragamonedas) es una política ε-greedy con un solo parámetro $\theta = \varepsilon$:

$\pi_\varepsilon(a) =
\begin{cases}
1-\varepsilon+\frac{\varepsilon}{K}, & \text{si } a = a^* \
\frac{\varepsilon}{K}, & \text{en otro caso}
\end{cases}$

Donde:

- $\varepsilon \in [0,1]$ es el único parámetro que controla la exploración
- a^* es la acción que actualmente parece ser la mejor
- K es el número total de acciones disponibles

Esta política elige la mejor acción conocida con probabilidad $1-\varepsilon+\frac{\varepsilon}{K}$, y explora eligiendo aleatoriamente cualquier acción con probabilidad $\frac{\varepsilon}{K}$.

PROF

Si $\varepsilon = 0$, siempre elegimos la mejor acción conocida (pura explotación).
Si $\varepsilon = 1$, elegimos aleatoriamente entre todas las acciones (pura exploración).

Por ejemplo, con 4 máquinas y $\varepsilon = 0.2$:

- La máquina que creemos mejor tiene 0.8 + 0.2/4 = 0.85 de probabilidad
- Cada otra máquina tiene 0.2/4 = 0.05 de probabilidad

Para aprender, simplemente ajustamos nuestro estimado de qué máquina es mejor basándonos en las recompensas recibidas. La belleza de esta política es su simplicidad: un solo parámetro controla directamente el balance entre exploración y explotación.

Ejemplo avanzado: Bandido multibrazo con softmax

Un problema clásico en RL es el "bandido multibrazo", donde un agente debe elegir repetidamente entre K máquinas tragamonedas (como en un casino), cada una con diferente probabilidad de recompensa desconocida. El objetivo es maximizar la ganancia total.

Una política parametrizada más sofisticada para este problema es la política softmax:

$\pi_\theta(a) = \frac{e^{\theta_a/\tau}}{\sum_{i=1}^{K} e^{\theta_i/\tau}}$

Donde:

- θ_a es un parámetro que representa la preferencia estimada para cada acción a
- τ es un parámetro de "temperatura" que controla la exploración

Inicialmente, podríamos fijar $\theta_a = 0$ para todas las acciones, dando igual probabilidad a cada máquina. Después de probar la máquina a y obtener una recompensa r, actualizamos:

$\theta_a \leftarrow \theta_a + \alpha(r - \bar{r})$

Donde α es la tasa de aprendizaje y $\bar{r}$ es la recompensa promedio observada hasta el momento.

Si la máquina 3 da una recompensa alta varias veces, su θ_3 aumentará, haciendo que la política la seleccione con mayor frecuencia. Al mismo tiempo, la temperatura τ puede disminuirse gradualmente para favorecer la explotación de las mejores máquinas identificadas. Este ejemplo muestra cómo una política parametrizada simple puede aprender de la experiencia sin necesidad de programar reglas específicas.

---

4. Retorno y Objetivo de Desempeño

El **retorno** desde el tiempo t es la recompensa acumulada con descuento:

$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$

PROF

El **objetivo de desempeño** $J(\theta)$ mide qué tan buena es una política, definido comúnmente como el retorno esperado desde la distribución de estados iniciales:

$J(\theta) = \mathbb{E}*{\tau\sim p\theta}[G_0] = \mathbb{E}*{S_0,A_0,S_1,\dots}\Big[\sum*{t=0}^{\infty} \gamma^t R_{t+1} \Big].$

El objetivo del RL es encontrar $\theta^* = \arg\max_\theta J(\theta)$.

---

5. Funciones de Valor

Las funciones de valor cuantifican el retorno esperado comenzando desde un estado (o par estado-acción) bajo la política π.

5.1 Función de Valor de Estado

$V^\pi(s) = \mathbb{E}\pi\big[G_t | S_t = s\big] = \mathbb{E}\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \big| S_t=s\Big].$

Mide qué tan bueno es estar en el estado s bajo la política π.

5.2 Función de Valor Acción-Estado

$Q^\pi(s,a) = \mathbb{E}_\pi\big[G_t | S_t = s, A_t = a\big].$

Mide qué tan bueno es tomar la acción a en el estado s bajo la política π.

5.3 Comparación intuitiva de funciones

Para entender mejor las diferencias y relaciones entre G_t, $J(\theta)$, $V^\pi(s)$ y $Q^\pi(s,a)$, pensemos en ellas de manera intuitiva:

Retorno (G_t): Es como el "puntaje final" real que obtuvo el agente a partir del tiempo t. Por ejemplo, si jugamos un videojuego, G_t sería la suma de todos los puntos que conseguimos desde ese momento hasta terminar el juego, aplicando descuento a los puntos futuros. Es un valor concreto para una ejecución específica, no un promedio.

Función de valor de estado ($V^\pi(s)$): Es como "¿Cuántos puntos espero conseguir si estoy en esta posición?". Si estamos en un mundo de cuadrícula, $V^\pi(s)$ nos dice cuánta recompensa total acumulada esperamos obtener si comenzamos en la casilla s y seguimos la política π hasta terminar. Es un promedio de todos los posibles G_t que podríamos obtener desde s.

Función de valor acción-estado ($Q^\pi(s,a)$): Es como "¿Cuántos puntos espero conseguir si estoy en esta posición y elijo este movimiento específico?". Nos dice cuánta recompensa total acumulada esperamos obtener si en el estado s tomamos la acción a y después seguimos la política π. Piensa en Q como "calidad" de un par estado-acción.

Objetivo de desempeño ($J(\theta)$): Es como "¿Qué tan buena es esta estrategia en general?". Mide qué tan bien le va a un agente que utiliza la política parametrizada π_θ en promedio, teniendo en cuenta todos los estados iniciales posibles.

PROF

Relaciones clave:

1. $V^\pi(s)$ es el promedio ponderado de $Q^\pi(s,a)$ sobre todas las acciones: $V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s,a)$
2. $Q^\pi(s,a)$ nos dice el retorno esperado después de tomar la acción a en el estado s: $Q^\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t=s, A_t=a]$
3. $J(\theta)$ es esencialmente $V^{\pi_\theta}(s_0)$ promediado sobre todos los estados iniciales posibles.

Ejemplo concreto (mundo de cuadrícula):

Imagina un mundo de cuadrícula de 3×3 con una meta en la esquina superior derecha que da +10 puntos. Cada paso cuesta -1 punto.

- G_t sería la suma de los puntos que obtienes en un recorrido específico. Por ejemplo, si tardas 3 pasos en llegar a la meta desde tu posición actual, $G_t = -1 -1 -1 + 10 = 7$.
- $V^\pi(s)$ nos diría el promedio de puntos que esperamos obtener desde cada casilla. Por ejemplo, la casilla justo a la izquierda de la meta tendría $V^\pi(s) \approx 9$ (un paso a la meta), mientras que la casilla más alejada tendría un valor menor.
- $Q^\pi(s,a)$ nos diría si es mejor movernos hacia arriba, abajo, izquierda o derecha desde cada posición. Por ejemplo, en la casilla debajo de la meta, $Q^\pi(s,\text{arriba})$ sería mayor que $Q^\pi(s,\text{derecha})$.
- $J(\theta)$ nos diría el desempeño promedio de nuestra política. Una política que siempre se mueve hacia la meta tendrá un $J(\theta)$ mayor que una que se mueve aleatoriamente.

Estas funciones se evalúan de diferentes maneras:

- G_t se calcula simplemente sumando las recompensas obtenidas.
- V^π y Q^π se pueden calcular mediante dinámica de programación, métodos de Monte Carlo o aprendizaje por diferencia temporal.
- $J(\theta)$ se optimiza mediante algoritmos de gradiente de política o métodos evolutivos.

El objetivo final del aprendizaje por refuerzo es encontrar la política óptima π^* que maximice cualquiera de estas medidas.

---

6. Ecuaciones de Bellman

Las ecuaciones de expectativa de Bellman relacionan las funciones de valor de forma recursiva.

6.1 Para V^π

$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) \big[R(s,a,s') + \gamma V^\pi(s') \big].$

6.2 Para Q^π

PROF

$Q^\pi(s,a) = \sum_{s'} P(s'|s,a) \big[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s',a') \big].$

Estas forman la base de muchos algoritmos de RL.

---

7. Intuición y Ejemplo

Consideremos nuestro mundo de cuadrícula:

- **Política**: moverse aleatoriamente. Entonces:
    - $V^\pi(s)$ da los pasos esperados (negativos) para alcanzar la meta desde cada celda.
    - $Q^\pi(s,a)$ indica qué acción es mejor en cada celda (por ejemplo, moverse hacia la meta).
- Cuando $\gamma\to1$, las recompensas a largo plazo importan más; cuando $\gamma\to0$, nos enfocamos en la recompensa inmediata.

Maximizar $J(\theta)$ mejora la política para alcanzar la meta más rápido con mayor recompensa acumulada.

---

8. Resumen de Símbolos Clave

Símbolo	**Definición**
$\mathcal{S}, \mathcal{A}$	Espacios de estados y acciones
$\pi(a	s)$
G_t	Retorno desde el tiempo t
$J(\theta)$	Objetivo de desempeño
$V^\pi(s)$	Función de valor de estado bajo política π
$Q^\pi(s,a)$	Función de valor acción-estado bajo política π
P,R,γ	Transición, recompensa, factor de descuento
