



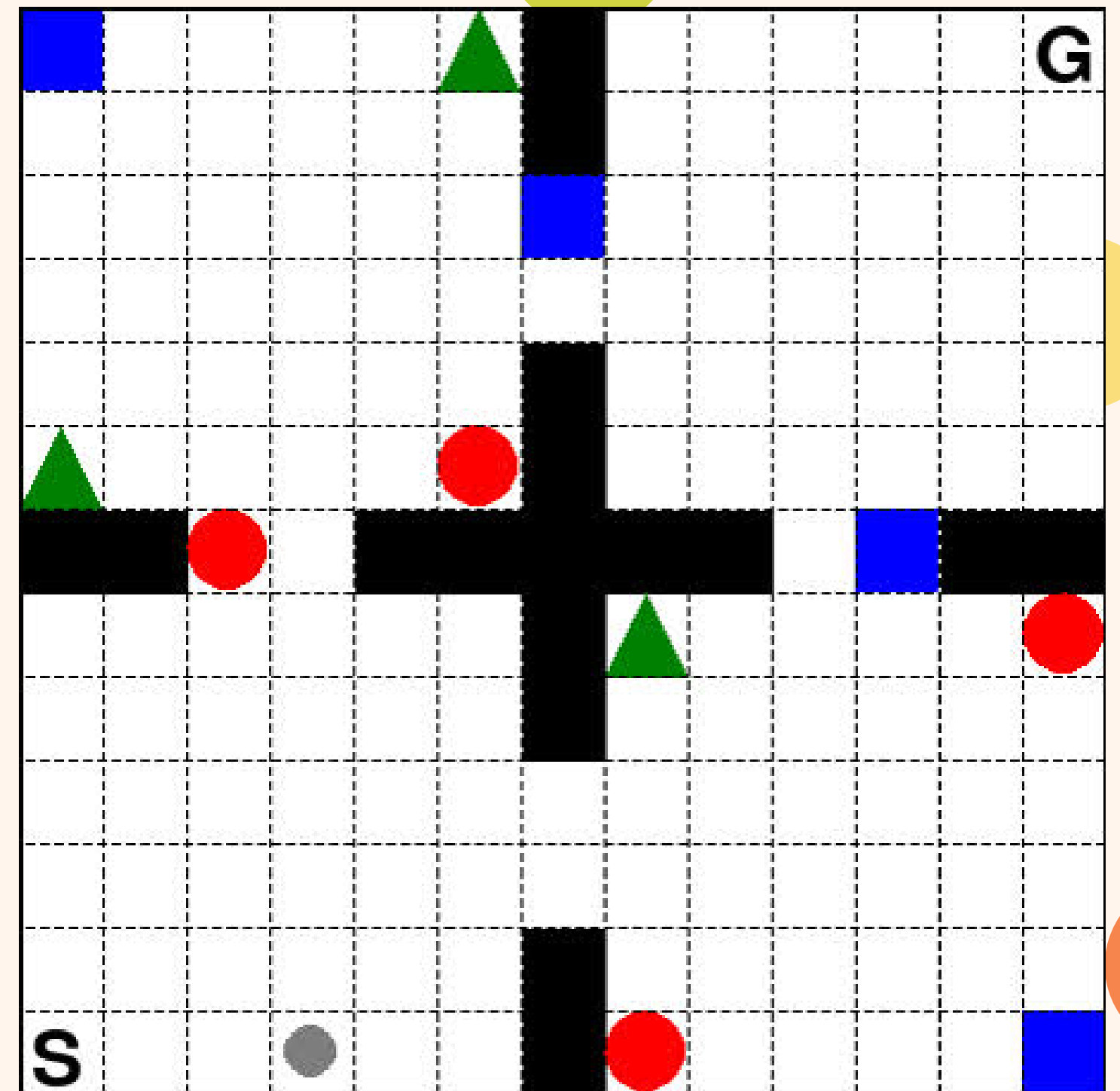
Aprendizaje por Refuerzo

FOUR-ROOM-VO

Presentado por :
Zurita Paco Elvis Jherson
Escobar Ruiz Marco Antonio

FOUR-ROOM

Aquí, un agente aprende a Recoge formas con recompensa positiva, y luego viaja a un objetivo fijo. El mundo de la cuadrícula está dividido en cuatro habitaciones separadas por paredes con pasadizos.



ARGUMENTOS

Espacio de observación :

La observación contiene la posición 2D del agente en el mundo de la cuadrícula, además de un vector binario que indica qué elementos se recogieron.

Espacio de acción :

El espacio de acción es discreto con 4 acciones: izquierda, arriba, derecha, abajo.

Espacio de recompensa :

La recompensa es un vector tridimensional con los siguientes componentes:

- +1 si recoges un cuadrado azul, de lo contrario 0
- +1 si recoge un triángulo verde, de lo contrario 0
- +1 si recoge un círculo rojo, de lo contrario 0

Estado inicial :

El agente comienza en la parte inferior izquierda del mapa.

Terminación del episodio:

El episodio termina cuando el agente alcanza el estado objetivo, G.

TABLA Q

En este entorno four-room, la observación obs es un vector de dimensión 14 que codifica, 2 valores para la posición (x, y) del agente dentro de la cuadrícula. y 12 valores binarios (0 o 1) que indican si se han recogido los diferentes elementos disponibles.

Acciones :son las estimaciones de la recompensa acumulada que el agente espera si, estando en el estado dado, toma esa acción y luego sigue su política óptima.

POSICIÓN

RECOMPENSAS

ACCIONES

Estado (12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0): [4 1 9 7]

Q-LEARNING ESTÁNDAR: LOS FUNDAMENTOS

Entrenar a un agente para aprender la mejor secuencia de acciones en un entorno (como un laberinto) con el objetivo de maximizar sus recompensas totales, utilizando el algoritmo Q-Learning en su forma más común.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (R + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a))$$

- $Q(s, a)$: El valor Q actual del par (estado, acción) que acaba de tomarse.
- α (alpha): Tasa de Aprendizaje (0.1)
- R (scalar_reward): Recompensa Inmediata por tomar la acción a en el estado s .
- γ (gamma): Factor de Descuento (0.99) Valora las recompensas futuras.
- $\max_{a'} Q(s', a')$ (best_next): Valor del Siguiente Estado máximo Q-value posible desde el estado siguiente s' .
- $(R + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a))$: Error TD La diferencia entre lo que el agente esperaba ($Q(s, a)$) y la "verdad" mejorada que acaba de experimentar o prever (td_target).

Q-LEARNING CON TASA DE APRENDIZAJE DECREMENTAL (INCREMENTAL)

Hacer que el agente confíe más en su conocimiento a medida que explora. La tasa de aprendizaje (α) disminuye cada vez que se repite un par (estado, acción).

TASA DE APRENDIZAJE ADAPTATIVA

$$\alpha_{adaptativo} = \frac{\alpha_{base}}{1 + k \cdot N(s, a)}$$

- α_{base} : La tasa de aprendizaje inicial (en el código, α).
- $N(s, a)$: Cuántas veces hemos tomado la acción a desde el estado s .
- k : Una constante pequeña que controla la velocidad de la disminución (en el código, 0.01).

Q-LEARNING CON INICIALIZACIÓN OPTIMISTA

Entrenar a un agente para aprender la mejor política en un entorno (como un laberinto) maximizando las recompensas, con un sesgo inicial hacia la exploración de lo desconocido.

- Inicio: El agente se encuentra en una casilla del laberinto (un estado). Como todos los Q-values iniciales son altos (1.0), el agente está dispuesto a probar cualquier dirección (acción).
- Movimiento: Digamos que el agente se mueve "arriba". El entorno le da una reward (probablemente pequeña o cero por no haber llegado a la meta aún) y lo coloca en next_state.
- Actualización (Fórmula): La fórmula de Q-learning se aplica. Si la acción fue neutral en recompensa, pero el next_state aún tiene valores optimistas para sus acciones, la $q_table[state][action]$ se actualizará ligeramente, pero el optimismo inicial seguirá siendo el principal motor. Si chocó con una pared, el reward negativo hará que el Q-value para esa acción en ese estado caiga drásticamente, aprendiendo rápido que es una mala idea.

Q-LEARNING CON SELECCIÓN DE ACCIÓN UCB (UPPER CONFIDENCE BOUND)

Entrenar a un agente que aprenda de manera eficiente en un entorno, balanceando de forma inteligente la exploración de acciones desconocidas y la explotación de las acciones que ya sabe que son buenas.

$$UCB(s, a) = Q(s, a) + c \cdot \sqrt{\frac{\ln(\text{total_visits}(s))}{\text{visit_count}(s, a) + \epsilon}}$$

- $Q(s, a)$: El valor Q actual de tomar la acción a desde el estado s . Representa la explotación (lo que el agente ya sabe).
- c : Un coeficiente de exploración ($c=2.0$). Controla la intensidad de la exploración. Un valor más alto significa más exploración.
- $\text{total_visits}(s)$: El número total de veces que el agente ha visitado el estado s .
- $\text{visit_count}(s, a)$: El número de veces que el agente ha tomado la acción a desde el estado s .
- ϵ ($1e-10$): Un valor pequeño para evitar divisiones por cero.

Q-LEARNING CON ADAPTACIÓN DE TASA DE APRENDIZAJE POR GRADIENTE

Entrenar a un agente para que aprenda una política óptima, utilizando una estrategia de selección de acciones probabilística (Softmax) y ajustando la velocidad de aprendizaje basándose en qué tan "sorprendido" está el agente (magnitud del error).

Selección de Acción Suave: Softmax en Lugar de ϵ -greedy

- `action = softmax_action(q_table[state], epsilon)`
- Enfoque: En lugar de elegir una acción aleatoria con probabilidad ϵ o la mejor conocida, Softmax asigna una probabilidad a cada acción basándose en su Q-value.

epsilon (temperatura): En Softmax, epsilon a menudo actúa como un parámetro de "temperatura".

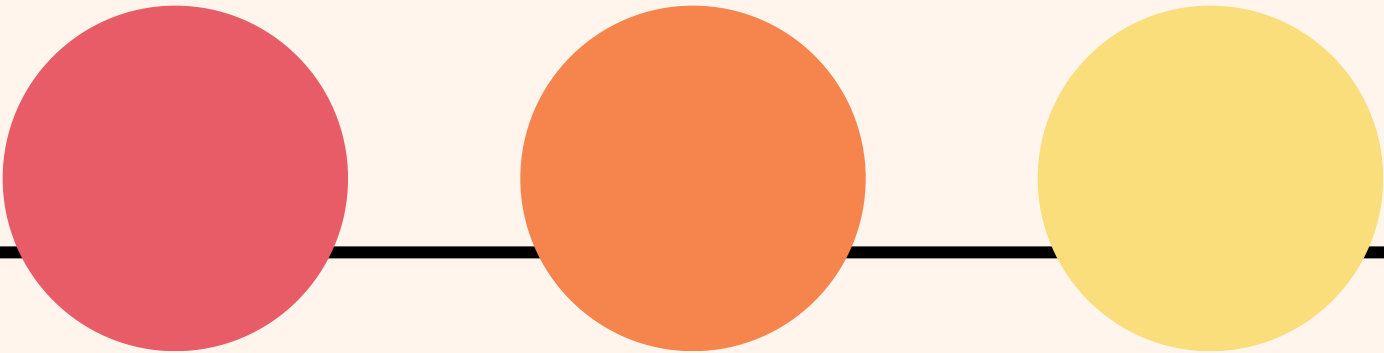
- Un epsilon alto (temperatura alta) hace que las probabilidades de todas las acciones sean más uniformes (más exploración).
- Un epsilon bajo (temperatura baja) hace que las acciones con Q-values altos sean mucho más probables (más explotación).



TASA DE APRENDIZAJE ADAPTATIVA: APRENDER AL RITMO DEL ERROR

El Error TD (td_error): La diferencia entre la recompensa esperada ($Q(s,a)$) y la recompensa real más el valor descontado del siguiente estado. Es una medida de la "sorpresa" del agente.

$$\text{td_error} = R + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)$$

- Ajuste del alpha (effective_alpha): `gradient_scale = np.tanh(abs(td_error))`
 - `effective_alpha = alpha * gradient_scale`
 - La función tangente hiperbólica (tanh) "aplasta" el valor absoluto del td_error en un rango entre 0 y 1.
- 

FUNCIÓN TRAIN_AND_COMPARE

Automatizar el proceso de entrenar, evaluar y comparar el rendimiento de todos los métodos de Q-learning que hemos definido. Sirve como el "orquestador" del experimento.

Entrenar Cada Agente: La función itera sobre este diccionario. Para cada método, ejecuta su función de entrenamiento, la cual devuelve la tabla Q aprendida y las recompensas por episodio.

Evaluar el Rendimiento: Después de que un agente es entrenado, se llama a `evaluate_policy` para medir su rendimiento de forma objetiva (sin exploración). Esto nos da la recompensa media y su desviación estándar.

GRACIAS

