UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA CENTRO UNIVERSITARIO DE QUETZALTENANGO CARRERA DE INGENIERIA EN SISTEMAS

PRINCIPIO DE CRITICA

ESTUDIANTE	CARNET:
MYNOR ESTUARDO BARAN LEIVA	1490-20-6399
DANIEL JOSUE FUENTES FIGUEROA	1490-20-3955
LUIS EDUARDO CAYAX PEREZ	1490-20-8439
ERICKON LEONARDO TACAM BATZ	1490-20-7748

24 de febrero 2024

Tipo de agente	Rendimiento	Entorno	Actuadores	Sensores
Aprendizaje por refuerzo	Llegar a la meta, lograr el objetivo en el menor tiempo, evitar los obstáculos	Laberinto, obstáculos	motores, llantas, servomotores	Camaras, sensores de proximidad

PSEUDOCÒDIGO

Inicializar estado del laberinto
Inicializar polítíca de acciòn del agente con valores iniciales

Mientras el agente no alcance la meta:
    observar el estado actual del entorno
    Si hay un evento estocàstico:
        Ajustar la estrategia basada en la probabilidad del evento
    Elegir una acciòn basada en la polìtica de acciòn actual
    Realizar la acciòn:
        Si la acciòn conduce a una recompensa, actualizar la política de acciòn
        Si la acciòn conduce a un obstáculo, ajustar la política para evitarlo
    Actualizar el estado del agente basado en la acciòn
    Aprender y ajustar la polìtica de acciòn utilizando el aprendizaje por refuerzo

Repetir hasta que el agente alcance la meta

REAS: Agente de Compra Automatica

Tipo de agente	Rendimiento	Entorno	Actuadores	Sensores
Comprador Acutomatico	información del mercado: precios, reseñas, ofertas especiales. cambio de precio.	Decisión de compra: comprar o no comprar.	historial de compras, preferencia de usuario, datos de precios y productos.	algoritmo de aprendizaje por refuerzo, evaluación de productos.

Entorno de Trabajo: Mercado Virtual

ENTORNOS DE TRABAJO	OBSERVABLE	DETERMINISTA	EPIDOSICO	ESTATICO	DISCRETO	AGENTES
MERCADO VIRUTAL	PARCIALMENTE	DETERMINISTAS	EPISODICO	ESTATICO	DISCRETO	MULTIPLES

TIPO DE AGENTES: agente basado en objetivos
el agente es del tipo ‘basado en objetivos’ porque el agente decide que productos comprar en función de la maximización de este objetivo y es el caso de nuestro agente, maximizar la satisfacción del usuario realizando compras dentro del presupuesto

Programa del Agente Algoritmo.

Clase: AgenteCompra

- **Atributos**:
  - presupuesto: La cantidad de dinero disponible para el agente para gastar en productos.
- **Métodos**:
  - __init__(self, presupuesto): Constructor de la clase, inicializa el agente con un presupuesto dado.
  - tomar_decision(self, productos): Método que simula la toma de decisiones del agente al comprar productos.

Algoritmo del Método tomar_decision:

1. Itera a través de cada producto en la lista de productos.
2. Genera un cambio de precio aleatorio entre -10% y +10% utilizando la función random.uniform(-0.1, 0.1).
3. Calcula el precio_actualizado del producto multiplicando su precio por 1 + cambio_precio.
4. Si el precio_actualizado es menor o igual al presupuesto del agente:

- Imprime un mensaje indicando la compra del producto y su precio actualizado.

- Reduce el presupuesto del agente por el precio_actualizado del producto comprado.

5. Si el precio_actualizado es mayor al presupuesto del agente:
- Imprime un mensaje indicando que no hay presupuesto suficiente para comprar producto.

```
import random #importamos la libreria de random para toma de deciones aleatorias
class AgenteCompra:
    def __init__(self, presupuesto):#metodo constructor
        self.presupuesto = presupuesto#establece el presupuesto
    def tomar_decision(self, productos):#funcion toma de decisiones
    # Simulamos la toma de decisiones basada en cambios de precios
        for producto in productos:#for para iterar toda la lista de productos
            cambio_precio = random.uniform(-0.1, 0.1) # Simula cambios de precio
            precio_actualizado = producto['precio'] * (1 + cambio_precio) #acutaliza el precio del poroducto

        if precio_actualizado <= self.presupuesto:#si el presupuesto es mayor entra
            print(f"Compra de {producto['nombre']} por ${precio_actualizado:.2f}")#imprime que compro el producto
            self.presupuesto -= precio_actualizado#se resta el precio del producto del presupuesto
        else:#si no alcanza el presupusto
            print(f"No hay presupuesto suficiente para {producto['nombre']}") #imprime que no hay presupuesto

# Ejemplo de uso
presupuesto_inicial = 100.0#presupuesto para el ejemplo
```

```
agente = AgenteCompra(presupuesto_inicial)#creamos el
agente con el presupuesto
productos_en_venta = [#creamos un array de productos.
 {'nombre': 'Producto A', 'precio': 20.0},
 {'nombre': 'Producto B', 'precio': 30.0},
 {'nombre': 'Producto C', 'precio': 15.0},
]

# Simulamos varias iteraciones para mostrar cómo el
agente ajusta sus decisiones
for _ in range(3):
    print("\nIteración:")
    agente.tomar_decision(productos_en_venta)
    print(f"Presupuesto restante:
${agente.presupuesto:.2f}")
```