UVG
UNIVERSIDAD DEL VALLE DE GUATEMALA

Reinforcement Learning
- Laboratorio 4 -

Semestre II - 2024

Instrucciones:

- Esta es una actividad en grupos de 3 personas máximo
- No se permitirá ni se aceptará cualquier indicio de copia. De presentarse, se procederá según el reglamento correspondiente.
- Tendrán hasta el día indicado en Canvas.

Task 1

Responda a cada de las siguientes preguntas de forma clara y lo más completamente posible.

1. ¿Cómo afecta la elección de la estrategia de exploración (exploring starts vs soft policy) a la precisión de la evaluación de políticas en los métodos de Monte Carlo?
    a. Considere la posibilidad de comparar el desempeño de las políticas evaluadas con y sin explorar los inicios o con diferentes niveles de exploración en políticas blandas.
2. En el contexto del aprendizaje de Monte Carlo fuera de la póliza, ¿cómo afecta la razón de muestreo de importancia a la convergencia de la evaluación de políticas? Explore cómo la razón de muestreo de importancia afecta la estabilidad y la convergencia.
3. ¿Cómo puede el uso de una soft policy influir en la eficacia del aprendizaje de políticas óptimas en comparación con las políticas deterministas en los métodos de Monte Carlo? Compare el desempeño y los resultados de aprendizaje de las políticas derivadas de estrategias épsilon-greedy con las derivadas de políticas deterministas.
4. ¿Cuáles son los posibles beneficios y desventajas de utilizar métodos de Monte Carlo off-policy en comparación con los on-policy en términos de eficiencia de la muestra, costo computacional. y velocidad de aprendizaje?

Task 2

En este ejercicio, simulará un sistema de gestión de inventarios para una pequeña tienda minorista. La tienda tiene como objetivo maximizar las ganancias manteniendo niveles óptimos de existencias de diferentes productos. Utilizará métodos de Monte Carlo para la evaluación de pólizas, exploring starts, soft policies y aprendizaje off-policy para estimar el valor de diferentes estrategias de gestión de inventarios. Su objetivo es implementar una solución en Python y responder preguntas específicas en función de los resultados.

Instrucciones:

1. Defina el entorno:
    a. Utilice el ambiente dado más adelante para simular el entorno de la tienda. Considere que:
        i. El estado representa los niveles de existencias actuales de los productos.
        ii. Las acciones representan decisiones sobre cuánto reponer de cada producto.
2. Generar episodios:
    a. Cada episodio representa una serie de días en los que la tienda sigue una política de inventario específica.
    b. Debe recopilar datos para varios episodios y registrar las recompensas (ganancias) de cada día.
3. Exploring Starts:
    a. Implemente explorar inicios para garantizar un conjunto diverso de estados y acciones iniciales.
4. Soft Policies:
    a. Utilice una soft policy (como epsilon-greedy) para garantizar un equilibrio entre la exploración y la explotación.
5. Aprendizaje off-policy:
    a. Implemente el aprendizaje off-policy para evaluar una política objetivo utilizando datos generados por una política de comportamiento diferente.

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Reinforcement Learning
- Laboratorio 4 -

Semestre II - 2024

Preguntas para responder:

1. ¿Cuál es el valor estimado de mantener diferentes niveles de existencias para cada producto?
2. ¿Cómo afecta el valor epsilon en la política blanda al rendimiento?
3. ¿Cuál es el impacto de utilizar el aprendizaje fuera de la política en comparación con el aprendizaje dentro de la política?

Entregas en Canvas

1. Documento PDF con las respuestas a cada task
    a. Pueden exportar el JN como PDF si trabajan con esto.
2. Código de la implementación del Task 2
    a. Si trabaja con JN deje evidencia de la última ejecución
    b. Caso contrario, deje en comentarios el valor resultante

Evaluación

1. [1 pts] Task 1 (0.25 cada pregunta)
2. [4 pts] Task 2

:

```
import numpy as np
import random

# Definicion de ambiente
class InventoryEnvironment:
    def __init__(self):
        self.products = ['product_A', 'product_B']
        self.max_stock = 10 # Pueden cambiar este número si gustan
        self.demand = {'product_A': [0, 1, 2], 'product_B': [0, 1, 2]}
        self.restock_cost = {'product_A': 5, 'product_B': 7}
        self.sell_price = {'product_A': 10, 'product_B': 15}
        self.state = None

    def reset(self):
        self.state = {product: random.randint(0, self.max_stock) for product in self.products}
        return self.state

    def step(self, action):
        reward = 0
        for product in self.products:
            stock = self.state[product]
            restock = action[product]
            self.state[product] = min(self.max_stock, stock + restock)
            demand = random.choice(self.demand[product])
            sales = min(demand, self.state[product])
            self.state[product] -= sales
            reward += sales * self.sell_price[product] - restock * self.restock_cost[product]
        return self.state, reward

# Init el ambiente
env = InventoryEnvironment()
```