UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Reinforcement Learning
- Laboratorio 8 -

Semestre II - 2024

Instrucciones:

- Esta es una actividad en grupos de 5 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 Único

Selecciona un ambiente de gymnasium y resuélvelo con DQN. Para esto, considere lo siguiente:

1. Elegir un ambiente de Gymnasium:
    - Selecciona un ambiente de **Gymnasium** que sea compatible con **DQN**. A continuación, algunos ejemplos recomendados:
        - CartPole-v1
        - Taxi-v3
        - MountainCar-v0
        - Acrobot-v1 o LunarLander-v2
2. Implementar el algoritmo DQN:
    - Implementa el algoritmo **Deep Q-Network (DQN)** para resolver el ambiente que seleccionaste.
    - Asegúrate de utilizar las técnicas clave de DQN, tales como:
        - **Experience Replay:** Almacena transiciones de experiencia en un buffer y reutilízalas para mejorar la eficiencia del entrenamiento.
        - **Target Network:** Utiliza una red de destino (target network) para estabilizar el entrenamiento al separar las actualizaciones del valor QQQ.
3. Consideraciones técnicas:
    - **Red Neuronal:** Define una red neuronal que actúe como aproximador de la función Q. La entrada de la red será el estado del ambiente y la salida debe ser el valor Q para cada acción.
        - **Consejo:** Para ambientes sencillos como CartPole-v1, una red con una o dos capas ocultas de 128 o 256 neuronas debería ser suficiente.
    - **Exploración vs. Explotación:** Implementa una política epsilon-greedy para balancear entre exploración y explotación.
        - Inicializa ϵ en un valor alto (0.9 o 1.0) y reduce gradualmente durante el entrenamiento.
4. Hiperparámetros importantes:
    - **Tasa de aprendizaje:** Un valor típico está entre 0.001 y 0.0001.
    - **Tasa de descuento (γ):** Determina cuánto valoras las recompensas futuras. Un valor común es 0.99
    - **Batch Size:** Selecciona un tamaño de batch adecuado (por ejemplo, 32 o 64 muestras) al actualizar la red.
    - **Tamaño del buffer de experiencia:** Un tamaño común es de 10,000 transiciones, aunque esto puede variar según el ambiente.
5. Entrenamiento y evaluación:
    - **Entrena el modelo:** Corre el entrenamiento por al menos 500-1,000 episodios o hasta que la tarea sea resuelta (según la definición del ambiente).
    - **Visualiza el progreso:** Muestra el promedio de recompensas por episodio para observar si el agente está aprendiendo.
6. Evaluación final:
    - Después de entrenar el DQN, evalúa el desempeño del agente ejecutando varios episodios sin explorar (usando solo la política aprendida) para ver cómo se comporta.
        - De esta parte graba un video de cómo se comporta el agente y presentalo en canvas

- **Consejo:** Si implementaste todo correctamente, deberías observar que el agente mejora con el tiempo y logra completar la tarea de manera eficiente.

Notas adicionales:

- **Librerías recomendadas:** Utiliza las librerías de Python como **NumPy**, **PyTorch** o **TensorFlow** para implementar las redes neuronales y manejar los cálculos matriciales.
- **Monitoreo del ambiente:** Puedes utilizar `gym.make('Ambiente').render()` para visualizar el ambiente durante el entrenamiento.
- **Aprendizaje inestable:** Si tu agente no está aprendiendo de manera estable, prueba ajustando la tasa de aprendizaje o el tamaño del batch.

Entregas en Canvas

1. Código de la implementación del Task
    a. Si trabaja con JN deje evidencia de la última ejecución
    b. Caso contrario, deje en comentarios el valor resultante
    c. Debe entregar el PDF y el JN
2. Video de la ejecución final del agente
3. **POR FAVOR, SI USAN REPOSITORIO TAMBIÉN SUBAN LA VERSIÓN PDF A CANVAS**

Evaluación

1. [2 pts] Código
2. [3 pts] Video