EL7021: Tarea 2

Profesor: Javier Ruiz del Solar
Auxiliar: Francisco Leiva
Ayudante: Javier Mosnaim

Abril, 2022

Requisitos

- Python $\geq$ 3.x
- NumPy
- Matplotlib
- PyTorch
- OpenAI Gym (`pip3 install gym`)

Fechas de entrega

Parte I (Avance)	:	12 de abril, hasta las 23:59
Parte II (Final)	:	19 de abril, hasta las 23:59

Parte I: Q-Learning

Descripción

Considere un automóvil que se mueve en un carril unidimensional, pero que se encuentra atrapado entre dos montañas. El objetivo es lograr que el automóvil supere la montaña de la derecha, no obstante, su motor no es lo suficientemente potente como para lograr esta hazaña simplemente acelerando. Usted debe encontrar una política que permita que el auto supere la montaña usando Q-Learning tabular.

Instrucciones

1. Describa formalmente el MDP que define al problema (espacio de estados, espacio de acciones, función de recompensa, y función de transición de estados).
2. Discretice los estados del ambiente.
3. Complete las funciones `select_action` y `update` del archivo `qlearning.py` de acuerdo a las instrucciones proporcionadas en el código base.
4. Muestre los resultados obtenidos (recompensa promedio y ratio de éxito) utilizando los parámetros del algoritmo entregados por defecto.

5. Implemente el decaimiento lineal de `epsilon` siguiendo las instrucciones proporcionadas en el archivo `qlearning.py`. Muestre nuevamente los resultados que obtiene al cambiar a `epsilon` a un valor inicial de 0.6. Comente.

Parte II: Deep Q-Network

Descripción

El problema consiste en encontrar una política que permita balancear un poste que se encuentra unido a un carro que se mueve en un carril unidimensional. Dicho poste se encuentra unido al carro a través de una articulación no actuada. Para encontrar esta política implementará Deep Q-Network (DQN).

Instrucciones

1. Deep Q-Network (I)

   En el archivo `deep_qnetwork.py`:

   1.1 Complete la clase `DeepQNetwork` de acuerdo a las instrucciones proporcionadas en el código base.

   1.2 Complete el constructor de la clase `DeepQNetworkAgent`.

   1.3 Complete la función `select_action`. Note que debe adicionalmente programar el decaimiento lineal del parámetro `epsilon`.

2. Replay Buffer.

   En el archivo `replay_buffer.py`:

   2.1 Complete el constructor de la clase `ReplayBuffer`.

   2.2 Complete la función `store_transition` para guardar una transición en el *buffer*.

   2.3 Complete la función `sample_transitions` para muestrear aleatoriamente un *batch* de transiciones del *buffer*.

   2.4 Pruebe su *replay buffer*. Muestre que guarda y muestrea transiciones de forma efectiva con un ambiente a su elección. Reporte sus resultados (*array shapes*, indexación en casos de borde). Use un *buffer* pequeño para estas pruebas.

3. Deep Q-Network (II)

   3.1 Complete la función `replace_target_network`. Esta función debe copiar los parámetros de Q-Network en Target Q-Network.

   3.2 Programe la función `update` dentro del archivo `deep_qnetwork.py`.

4. Experimentos

   Para esta sección, fije los siguientes parámetros: `nb_training_steps:20000`, `gamma:0.99`, `epsilon:0.8` y `lr:0.01`.

   4.1 Pruebe su algoritmo con las siguientes combinaciones de parámetros:

ID Experimento	replay_buffer_size	batch_size	nb_steps_target_replace
exp_11	1000	16	100
exp_21	2500	16	100
exp_31	5000	16	100
exp_21	5000	32	100
exp_22	5000	64	100
exp_23	5000	128	100
exp_31	5000	128	1
exp_32	5000	128	100
exp_33	5000	128	1000

Cada experimento debe ser ejecutado cinco veces. Por cada experimento, reporte los resultados obtenidos en una figura que muestre los gráficos de recompensa y tasa de éxito promedios, junto a su desviación estándar. En su entrega, adjunte los archivos .csv con los resultados que usó para generar los gráficos de su reporte. Adjunte también el código empleado para cargar, procesar y graficar estos datos.

4.2 Analice los resultados obtenidos en los experimentos realizados, en términos de los parámetros empleados en cada experimento.

Reglas de formato

Las entregas deben cumplir con los siguientes requerimientos:

- Reporte en formato PDF.
- Incluya un archivo README.txt junto al código, donde indique las versiones de las dependencias que utilizó, y las instrucciones de ejecución de su código.
- Entregas parciales y finales en formato zip (reporte y código en un único archivo).
- Figuras legibles, de preferencia vectorizadas.
- Enumerar las respuestas de la misma forma en que se enumeran las preguntas.
- Respuestas concisas. No es necesario describir gráficos, es suficiente con mostrarlos en el reporte, y responder las preguntas que se realizan textualmente.
- No es necesario crear una portada, no obstante, la primera página debe contar con:
  - Título con formato “Avance Tarea X o Entrega Tarea X”, según corresponda.
  - Código del curso.
  - Nombre del estudiante.