EL7021: Tarea 3

Profesor: Javier Ruiz del Solar
Auxiliar: Francisco Leiva
Ayudante: Rodrigo Salas

Abril, 2023

Requisitos

- Python $\geq$ 3.x
- NumPy
- Matplotlib
- PyTorch
- OpenAI Gym (`pip install gym==0.23.1`)

Fechas de entrega

Parte I (Avance)	:	8 de mayo, hasta las 23:59
Parte II (Final)	:	15 de mayo, hasta las 23:59

Parte I

Descripción

Implementará un algoritmo *policy gradient*: la parametrización de la política, el muestreo de trayectorias, la estimación de retornos, y lo extenderá utilizando métodos simples de reducción de varianza. Probará su algoritmo en dos ambientes (uno con acciones discretas y otro con acciones continuas) y evaluará su desempeño variando algunos parámetros.

Instrucciones

1. Parametrización de la política.

   1.1 Complete la clase `Policy` en el archivo `policy_gradients.py` de acuerdo a las instrucciones proporcionadas en el código base.

   1.2 Complete la función `select_action` del archivo `policy_gradients.py`.

2. Muestreo de trayectorias.

   2.1 Complete la función `perform_single_rollout` del archivo `train_agent.py`. Verifique su funcionamiento usando el ambiente `CartPole` y `Pendulum`, y compruebe que las dimensiones de su salida son las especificadas en el código base.

2.2 Complete la función `sample_rollouts` del archivo `train_agent.py`. Nuevamente verifique que funcione con los ambientes `CartPole` y `Pendulum`.

3. Estimación de retornos.

   3.1 Complete la función `estimate_returns` del archivo `policy_gradients.py` de acuerdo a las instrucciones proporcionadas en el código base.

   3.2 Verifique el funcionamiento de su código muestreando a lo más dos trayectorias y estimando los retornos correspondientes. Para ello use tanto el ambiente `CartPole` como `Pendulum`. Reporte los resultados obtenidos (muestre los retornos que entrega su programa y justifique su validez).

4. *Policy gradients*.

   4.1 Programe la función `update` del archvio `policy_gradients.py`.

5. Reducción de varianza.

   5.1 Extienda la función `estimate_returns` del archivo `policy_gradients.py` para calcular el "*reward to go*".

   5.2 Extienda la función `estimate_returns` del archivo `policy_gradients.py` para calcular un *baseline* empleando los retornos obtenidos en un muestreo de trayectorias dado. Use este término para modificar la estimación de retornos.

6. Evaluación del algoritmo.

   6.1 Entrene un agente en el ambiente `CartPole` usando `training_iterations:100`, `lr:0.005`, `gamma:0.99`, y las siguientes combinaciones de parámetros:

ID Experimento	batch_size	use_baseline	reward_to_go
exp_11	500	False	False
exp_21	500	False	True
exp_31	500	True	False
exp_41	500	True	True
exp_12	5000	False	False
exp_22	5000	False	True
exp_32	5000	True	False
exp_42	5000	True	True

   Reporte sus resultados en un gráfico de recompensas por episodio promedio (por cada experimento), considerando tres ejecuciones para cada combinación de parámetros. Adjunte los archivos .csv con los resultados que obtenga, y también el código empleado para cargar, procesar y graficar estos datos.

   6.2 Entrene un agente en el ambiente `Pendulum` usando las mismas configuraciones que las utilizadas para el ambiente `CartPole`, pero variando el número de iteraciones (es decir, el parámetro `training_iterations`) a `1000`, y solo para un `batch_size` de tamaño `5000`.

   6.3 Analice los resultados obtenidos para el ambiente `CartPole`. En particular, comente las variaciones en los resultados obtenidos al (i) modificar el tamaño de los *batches*, (ii) usar o no usar *reward to go*, y (iii) usar o no usar un *baseline*.

   6.4 Realice el mismo análisis de la parte anterior para el ambiente `Pendulum` en función de los resultados obtenidos.

Parte II

Descripción

Implementará un algoritmo de tipo actor-crítico *on-policy*: la parametrización para el actor y el crítico, el muestreo de trayectorias, las reglas de actualización para el actor y el crítico, y pondrá a prueba el algoritmo abordando problemas de control clásicos.

Instrucciones

1. Parametrización del actor.

   1.1 Complete la clase `Actor` en el archivo `actor_critic.py`.

   1.2 Complete la función `select_action`.

2. Muestreo de trayectorias.

   2.1 Complete la función `perform_single_rollout` del archivo `train_agent.py`. Verifique su funcionamiento usando el ambiente `CartPole` y `Pendulum` y compruebe que las dimensiones de su salida son las especificadas en el código base.

   2.2 Complete la función `sample rollouts` del archivo `train_agent.py`. Nuevamente verifique que funcione con los ambientes `CartPole` y `Pendulum`.

3. Parametrización del crítico y actualización.

   3.1 Complete la clase `Critic` en el archivo `actor_critic.py`.

   3.2 Programe las funciones `update_actor` y `update_critic` siguiendo las instrucciones proporcionadas en el código base.

4. Experimentos.

   4.1 Entrene en el ambiente `CartPole` usando `actor_lr:0.001`, `critic_lr:0.001`, `gamma:0.99`, `training_iterations:200`, y las siguientes combinaciones de parámetros:

ID Experimento	batch_size	nb_critic_updates
exp_11	500	1
exp_21	500	10
exp_31	500	100
exp_12	5000	1
exp_22	5000	10
exp_32	5000	100

   Reporte sus resultados en un único gráfico de recompensas por episodio promedio (cada gráfico debe tener tres curvas). Adjunte los archivos .csv con los resultados que obtenga, y también el código empleado para cargar, procesar y graficar estos datos.

5. Entrene en el ambiente `Acrobot` usando las mismas configuraciones que las utilizadas para `CartPole` y reporte los resultados obtenidos.

6. Entrene en el ambiente `Pendulum` usando las mismas configuraciones que las utilizadas para el ambiente `CartPole`, pero variando el número de iteraciones (`training_iterations`) a `2000`, y solo para `batch_size` de tamaño `5000` (un único gráfico con tres curvas). Repita lo anterior, esta vez usando `critic_lr: 0.01`.

7. Analice los resultados obtenidos para el ambiente `CartPole` y `Acrobot`. En particular, comente las variaciones observadas al (i) modificar el tamaño de los *batches*, y (ii) modificar el número de actualizaciones del crítico.

8. Realice el mismo análisis de la parte anterior para el ambiente `Pendulum` en función de los resultados obtenidos.

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.