Ejercicios Lab1

October 21, 2018

Para el presente lab vamos a utilizar la clase FrozenLakeAgent, llamada desde el presente notebook, y el script *frozenlake_main_script*, ubicado en *agents/frozen_lake_agent/*.
Los mismos se presentan como herramientas para resolver los ejercicios, por lo que se permite, para la resolución de los mismos, modificarlos o reemplazarlos por sus propias implementaciones.
Presentan la siguiente funcionalidad:

frozenlake_main_script

Script que crea y define la configuración inicial del agente de RL de tipo FrozenLake, creando la instancia de FrozenLakeAgent.

FrozenLakeAgent

Clase que implementa la interfaz con OpenAIGym, creando el entorno FrozenLake y aplicando acciones sobre el mismo.
Adicionalmente provee una interfaz para llamar y correr el agente RL.

Adicionalmente, en la clase también se implementa el algoritmo QLearning, lo cual involucra el guardado de los valores de Q en un diccionario, la selección de acciones (mediante *choose_action*) y la actualización de los valores de Q (mediante *learn*).

Se pide:

1. Dado el agente de Q-Learning en el entorno FrozenLake, implementar la política Softmax, dada por
$$\pi(a \mid s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}$$

2. Realizar una breve descripción analizando cómo difieren en la curva de aprendizaje los distintos valores de los híper-parámetros α, ϵ, τ y γ tanto con política ϵ-greedy como con Softmax.

3. Implementar algoritmo SARSA y compararlo con Q-Learning, en la curva de convergencia de la recompensa.
Opcional: compararlos también en la matriz de valor, para lo cuál debe adaptarse la matriz de valor a la política Softmax (actualmente la misma está de acuerdo a la política epsilon-greedy).

4. Evaluar cómo cambia el desempeño de los algoritmos implementados con el entorno FrozenLake con Slippery = True.
£Cómo se comparan Q-Learning y SARSA? £Cómo es esta comparación de acuerdo a los valores de sus híper-parámetros?

5. Modificar la función de recompensa de modo tal que se penalice caer al agua o bien que se penalice cada time-step (o algún criterio similar).
   Analizar cómo difiere la convergencia en tales casos (comparando Q-Learning con SARSA).

6. Aumentar la cantidad de pasos posibles en el entorno (mediante el cutoff_time en el script y con max_episode_steps en la clase Q-Learning).
   Evaluar si la matriz de valor se ve afectada, y en tal caso describir cómo.

Para enviar el Lab, enviar la solución en un archivo comprimido a la dirección jbarsce | at | gmail | dot | com

Nota 1: puede utilizarse la versión más grande del FrozenLake, que es una grilla de tamaño 8x8 (solamente esta grilla o ambas grillas a modo de comparación).
Para usarla, se debe modificar la variable en *register* del método *init_agent* de '4x4' a 8x8.

Nota 2: para visualizar los distintos aspectos del entorno frozen lake que se pueden cambiar en la función Register, seguir este link https://github.com/openai/gym/blob/master/gym/envs/toy_text/frozen_lake.py

Recomendación general: No se sugiere hacer este TP desde jupyter notebook/lab sino desde un IDE estilo Pycharm, debido a que los algoritmos de RL suelen requerir un debug paso a paso, tanto para corregir errores como para entender mejor cómo funcionan los mismos.