EJERCICIOS de APRENDIZAJE por REFUERZO

M08 – DC03 – REINFORCED LEARNING

Programa: Máster en Big Data, Cloud & Analytics

Periodo académico: 2019 – 2020

Autor/es: CARLOS ALFONSEL JAÉN
carlos.alfonsel@mbitschool.com

1. Describe dos problemas diferentes que no se hayan dado en las sesiones de RL que se puedan enfocar con **MULTIARMED BANDITS NO CONTEXTUALES**. Di cuál es el agente, el entorno, las posibles acciones y los refuerzos.

Un ejemplo de Aprendizaje Reforzado No Contextual podría ser un robot limpiador (ver imagen). Las versiones más modernas de estos robots son capaces de mapear la superficie a limpiar dentro de un domicilio, mediante una técnica de prueba-y-error. El objetivo es optimizar el tiempo que se emplea en la limpieza, con el consiguiente ahorro de energía. Los primeros modelos (y hablo por experiencia), menos inteligentes, podían realizar múltiples pasadas por la misma zona de la casa y dejar otras sin pasar por ellas.

- **Agente**: el robot limpiador (Roomba).
- **Entorno**: la casa o habitación a limpiar.
- **Acciones posibles**: moverse hacia delante, retroceder, girar X grados en sentido horario.
- **Refuerzos**: positivo si avanza sin encontrar obstáculo, negativo si se encuentra un obstáculo y debe retroceder.

Otros ejemplos de problemas que se pueden enfocar desde una aproximación de Multiarmed Bandits No-Contextuales podrían ser los **sistemas recomendadores** de aplicaciones como **Twitter**, **IMDb**, o **Spotify**. En todos estos casos, el agente es el sistema automático de recomendación, el entorno es la aplicación correspondiente, y el refuerzo será positivo si el usuario finalmente accede al ítem recomendado.

2. Haz lo mismo con otros dos problemas que se puedan enfocar con **MULTIARMED BANDITS CONTEXTUALES**.

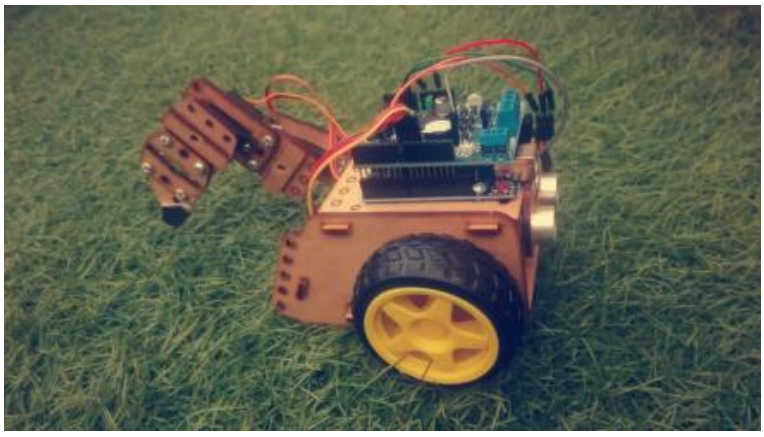
Un primer ejemplo de Aprendizaje Reforzado Contextual podemos verlo en la película de 1983 **JUEGOS de GUERRA** (*WarGames* en el original), dirigida por John Badham e interpretada por Matthew Broderick. En la sinopsis de IMDb podemos leer lo siguiente: *"A young man finds a back door into a military central computer in which reality is confused with game-playing, possibly starting World War III"*. Aún a riesgo de hacer un spoiler, en el desenlace de la película vemos cómo el súper-computador encargado de

la defensa estratégica de Estados Unidos, después de realizar millones de simulaciones de guerra nuclear, llega a la conclusión de que es imposible que ninguno de los dos contendientes gane, ya que en cualquier caso el mundo quedaría destruido.

- **Agente**: el súper-computador ("Joshua").
- **Entorno**: el mundo (simulado).
- **Acciones posibles**: lanzar o no misiles termonucleares, empezando así una guerra mundial.
- **Refuerzos**: positivo si ganas la guerra, negativo si la pierdes. Al final la conclusión a la que se llega es que la única forma de ganar es "no jugando".

Otros problemas que se podrían enfocar desde una aproximación Multiarmed Bandits Contextuales son aquellos **videojuegos de acción/rol**, donde los enemigos "aprenden" la forma de comportarse del usuario al enfrentarse a enemigos, y son capaces de ir modificando su estrategia para ir aumentando la dificultad o los desafíos del personaje. Algunos ejemplos podrían ser **Dark Souls III**, **The Witcher III: Wild Hunt**, la serie **God of War**, etc. Generalmente el usuario puede configurar el nivel de la IA detrás de ese aprendizaje, dependiendo de la dificultad en la que se quiera jugar. En estos casos, el agente es la propia IA del viadeojuego, el entorno es el "campo de batalla" donde se produce el enfrentamiento, las acciones dependerán del tipo de juego, y el refuerzo será positivo si se consigue acabar con el jugador y cero o negativo si se pierde.

3. Haz lo mismo con otros dos problemas que se puedan enfocar con Q-LEARNING. Además de decir cuáles serían las acciones posibles y refuerzos, di de qué orden sería el gamma.



Rastreando la web en busca de ejemplos de problemas de aprendizaje reforzado con Q-learning, he encontrado este proyecto: [Q-learning con Arduino: Crawling Robot]

Un robot rastreador es básicamente un robot que

utiliza el algoritmo Q-learning para poder "aprender" los mejores movimientos para trasladarse. Según se puede leer en este artículo, el **agente** (el robot) puede hacer hasta un total de 64 **acciones** distintas (16 posiciones posibles para el brazo, y 4 acciones por servomotor), para moverse en el entorno definido. Las **recompensas** que se definen van de 1 a 10, dependiendo de la acción realizada y de si el robot se ha movido al menos 2 cm. El **gamma** se ha configurado igual a 0.8.

Otros ejemplos de aplicación de una aproximación Q-learning serían todos aquellos relacionados con el movimiento autónomo de vehículos o dispositivos en entornos controlador. Por ejemplo, los robots encargados del movimiento de mercancías en una factoría o en un almacén (la imagen corresponde a uno de los **almacenes robotizados** de Amazon).

4. Haz una versión del notebook **2-catch-viento.ipynb** (carpeta **2-sequential_reinforcement_learning/05-catch**) donde uses la librería **gym** para definir el entorno. Ayuda: se hace eso mismo en el notebook **02-QL_treasure_on_right_gym.ipynb** de la carpeta **01-QL_treasure_on_right**.

Ver notebook adjunto **ejercicio_catch-viento_gym_keras-rl.ipynb**

5. **Opcional:** introduce además la librería **keras-rl** para entrenar tu modelo, de manera similar a lo que se hace en el notebook **02-dqn_cartpole_keras_rl.ipynb** a partir del **01-dqn_cartpole.ipynb**, ambos de la carpeta **04_DQN_cartpole**.

Ver notebook adjunto **ejercicio_catch-viento_gym_keras-rl.ipynb**

En este apartado sólo se plantea teóricamente la solución, puesto que al ejecutarse se produce un error que no he conseguido resolver:

```
# Training the model:

dqn.fit(env_keras, nb_steps = 10000, visualize = True, verbose = 2);
```

```
Training for 10000 steps ...

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-19-308a9189f409> in <module>
      1 # Training the model:
      2
----> 3 dqn.fit(env_keras, nb_steps = 10000, visualize = True, verbose = 2);

~/anaconda3/envs/env_rl/lib/python3.6/site-packages/rl/core.py in fit(self, env, nb_steps, action_repetition,
callbacks, verbose, visualize, nb_max_start_steps, start_step_policy, log_interval, nb_max_episode_steps)
    133                     if self.processor is not None:
    134                         observation = self.processor.process_observation(observation)
--> 135                     assert observation is not None
    136
    137                     # Perform random starts at beginning of episode and do not record them into the ex
perience.

AssertionError:
```

He definido el entorno con la librería **gym**, he creado la función **step** y el modelo con **keras-rl**, pero al ejecutar se produce ese error que no consigo solucionar.

```
def step(self, action):
    # This is how agent will interact with the environment
    self._update_state(action)
    reward    = self._get_reward()
    game_over = self._is_over()
    return self.observe(), reward, game_over
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_5 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 303
=================================================================
Total params: 20,503
Trainable params: 20,503
Non-trainable params: 0
_________________________________________________________________
None
```