Reinforcement Learning - QLearning

Juan Fernando Castañeda

Índice

1. Generalidades

Para todos los modelos se utilizó Value Iteration, pues recalcular la política cada episodio hubiera sido pesado computacionalmente. A cambio, el QValor estaba directamente disponible.

Para aumentar la rapidez con la que convergía se dio un valor negativo a cada paso que no estuviese ocupado por otro evento.

Toda la información de los ambientes está en el archivo **environments.py**. Los muros fueron implementados como casillas por las que no se podía pasar. Se parecen mucho los dos ambientes definidos (Env, y TaxiEnv), pero con sutiles diferencias para incluir nuevas acciones y mecánicas en TaxiEnv.

2. Gridworld

Para el modelo se hicieron varios experimentos con los parámetros α, γ y ϵ. Los más significativos, sin embargo, fueron los cambios en ϵ.

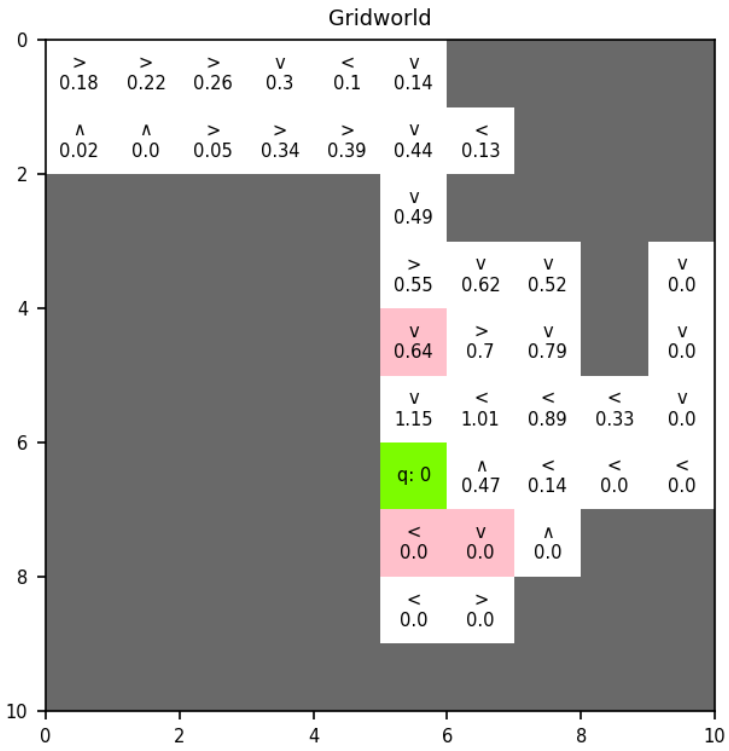
Para modificar los parámetros y correr el código se pueden modificar las siguientes líneas del archivo **agent_grid.py**. Específicamente, QLearning recibe como parámetros α, γ y ϵ.

```
138   agent1 = QLearning(0.1, 0.9, 0.9, gridworld)
139   print(f"Episode number Gridworld: {agent1.control()}")
140   agent2 = QLearning(0.1, 0.9, 0.9, labyrinth)
141   print(f"Episode number Labyrinth: {agent2.control()}")
142
143   agent1.plot_scenario("Gridworld")
144   agent2.plot_scenario("Labyrinth")
```

Gridworld parámetros.

2.1. $\epsilon = 0.1$

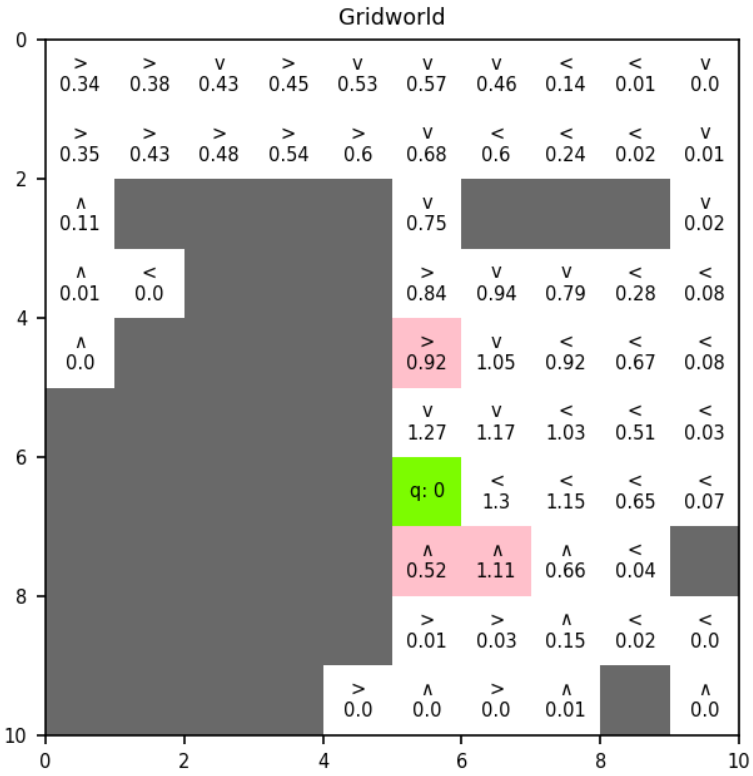
Con $\epsilon = 0.1$ tenemos que el agente no exploró en su totalidad los estados antes de converger. La cantidad de iteraciones que demoró antes de converger fueron 179. Precisamente por su poca exploración no tiene una política óptima en muchos puntos ya explorados. Por ejemplo, la política en los estados cercanos a los dos negativos de la fila 7.



Gridworld $\epsilon = 0.1$.

2.2. $\epsilon = 0.5$

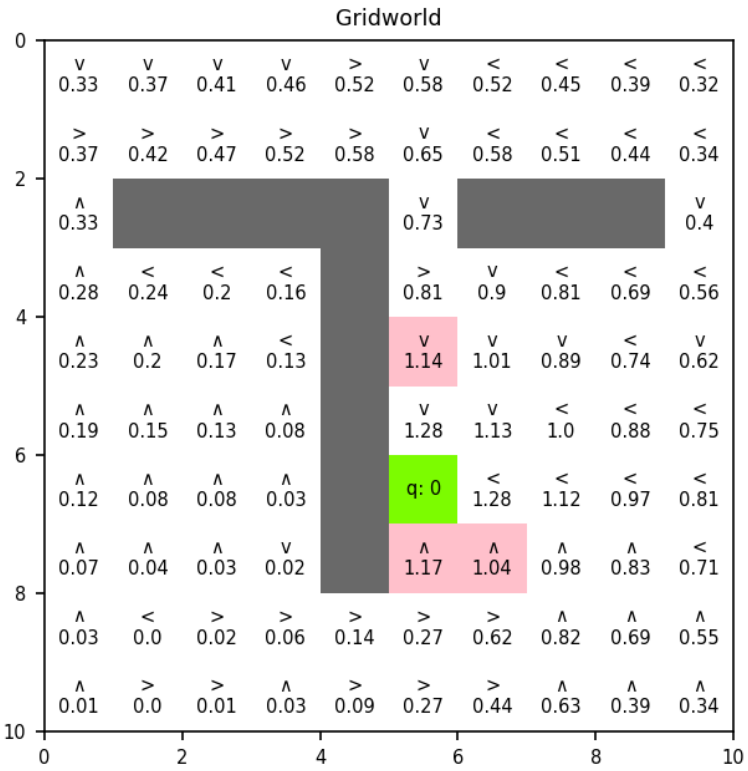
Con $\epsilon = 0.5$ tenemos que el agente tampoco exploró en su totalidad los estados antes de converger. La cantidad de iteraciones que demoró antes de converger fueron 301. Por su exploración media tiene política óptima en sus puntos ya explorados.



Gridworld $\epsilon = 0.5$.

2.3. $\epsilon = 0.9$

Con $\epsilon = 0.9$ tenemos que el agente exploró en su totalidad los estados antes de converger. La cantidad de iteraciones que demoró antes de converger fueron 242. Como exploró todo, pero no lo suficiente en algunos estados, hay algunas políticas que no son ideales. Sobre todo en la parte inferior izquierda, donde podría llegar de forma más directa a la solución. Se intentaron epsilons más altos para mejorar la precisión y funcionó, pero aumentar epsilon aún más implica volver la convergencia más demorado, pues se asejema cada vez más a un random walk.



Gridworld $\epsilon = 0.9$.

3. Labyrinth

Para el modelo se hicieron varios experimentos con los parámetros α, γ y ϵ. Los más significativos, sin embargo, fueron los cambios en ϵ.

Para modificar los parámetros y correr el código se pueden modificar las siguientes líneas del archivo **agent_grid.py**. Específicamente, QLearning recibe como parámetros α, γ y ϵ. Es el mismo lugar donde se modificaban los parámetros de Gridworld.

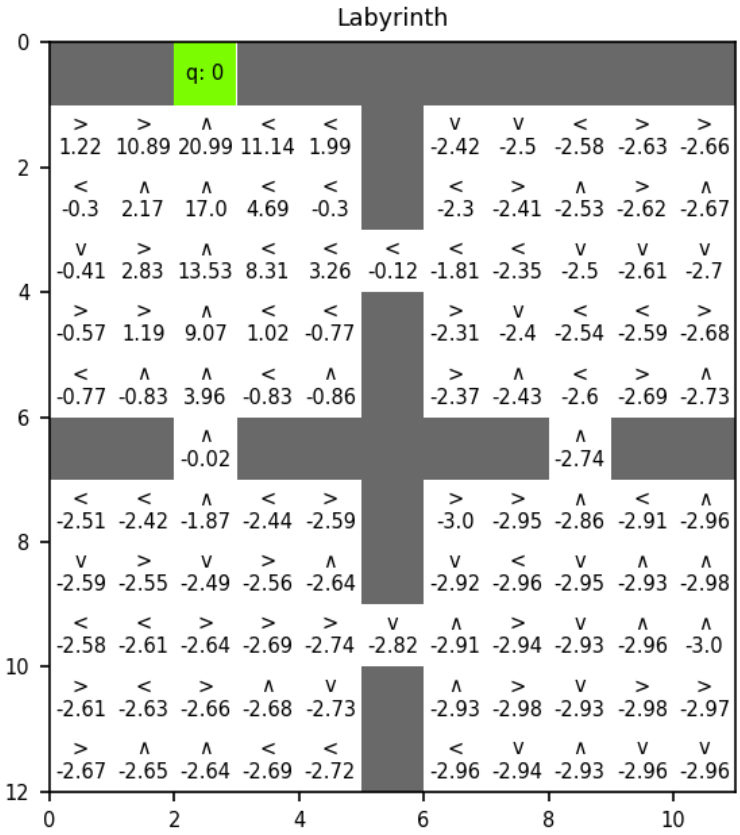
```
138   agent1 = QLearning(0.1, 0.9, 0.9, gridworld)
139   print(f"Episode number Gridworld: {agent1.control()}")
140   agent2 = QLearning(0.1, 0.9, 0.9, labyrinth)
141   print(f"Episode number Labyrinth: {agent2.control()}")
142
143   agent1.plot_scenario("Gridworld")
144   agent2.plot_scenario("Labyrinth")
```

Labyrinth parámetros.

3.1. $\epsilon = 0.1$

Con $\epsilon = 0.1$ tenemos que el agente no exploró lo suficiente todos los estados antes de converger. La cantidad de iteraciones que demoró antes de converger fueron 82. Precisamente por su poca exploración no tiene una política óptima en muchos puntos ya explorados. Por ejemplo, la política

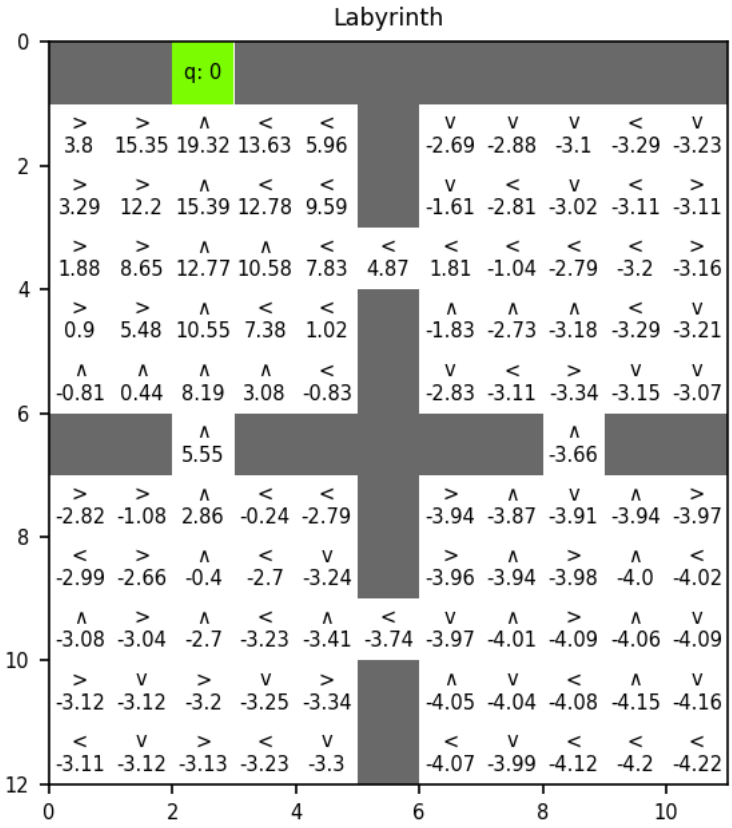
en los estados cercanos de la esquina inferior derecha, esquina inferior izquierda y esquina superior derecha no son óptimos. Incluso en la esquina superior izquierda hay estados sin política óptima.



Labyrinth $\epsilon = 0.1$.

3.2. $\epsilon = 0.5$

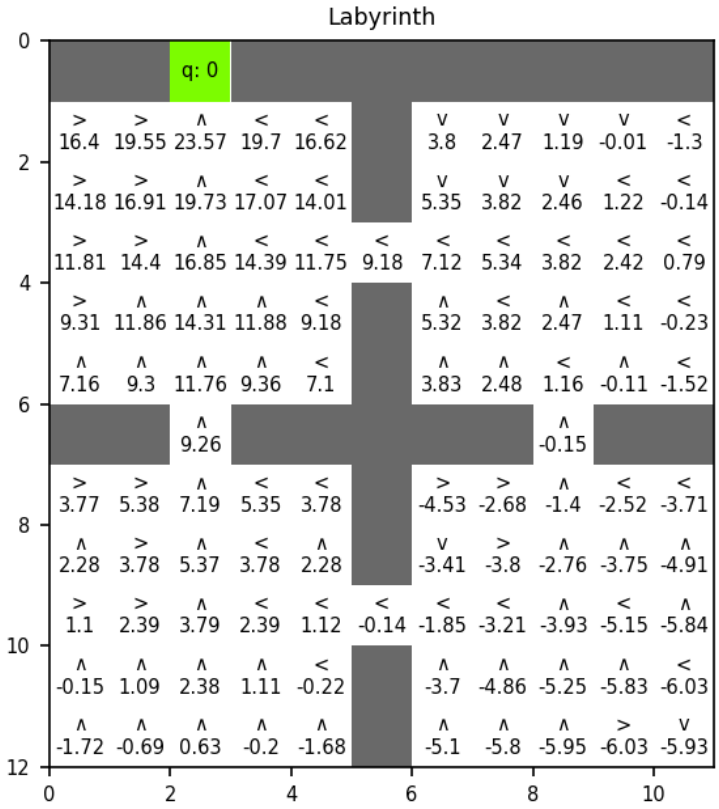
Con $\epsilon = 0.5$ tenemos que el agente no exploró lo suficiente todos los estados antes de converger. La cantidad de iteraciones que demoró antes de converger fueron 126. Por su exploración media no tiene una política óptima en muchos puntos ya explorados. Por ejemplo, la política en los estados cercanos de la esquina inferior derecha, y la mitad de los estados de la esquina inferior izquierda y esquina superior derecha, no son óptimos. A cambio, todos los estados de la esquina superior izquierda son óptimos.



Labyrinth $\epsilon = 0.5$.

3.3. $\epsilon = 0.9$

Con $\epsilon = 0.9$ tenemos que el agente exploró lo suficiente todos los estados antes de converger. La cantidad de iteraciones que demoró antes de converger fueron 223. Por su exploración alta tiene una política óptima en todos los puntos. Al igual que en el caso anterior, subirlo más podría implicar subir la complejidad computacional.



Labyrinth $\epsilon = 0.9$.

4. Taxi

Para el modelo se hicieron varios experimentos con los parámetros α, γ y ϵ. Los más significativos, sin embargo, fueron los cambios en ϵ.

Para modificar los parámetros y correr el código se pueden modificar las siguientes líneas del archivo **agent_taxi.py**. Específicamente, QTaxi recibe como parámetros α, γ y ϵ.

```
141    agent = QTaxi(0.1, 0.9, 0.9, taxi_env)
142    print(f"Episode number: {agent.control()}")
143    agent.plot_scenario(0)
144    agent.plot_scenario(1)
```

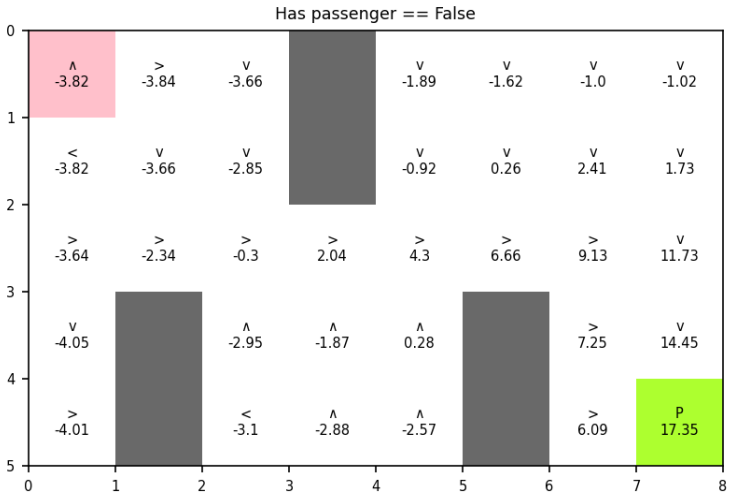
Taxi parametros.

El estado marcado con rojo es el elegido como dropoff y el marcado con verde es el elegido como pickup. Son diferentes cada vez que se inicia el programa.

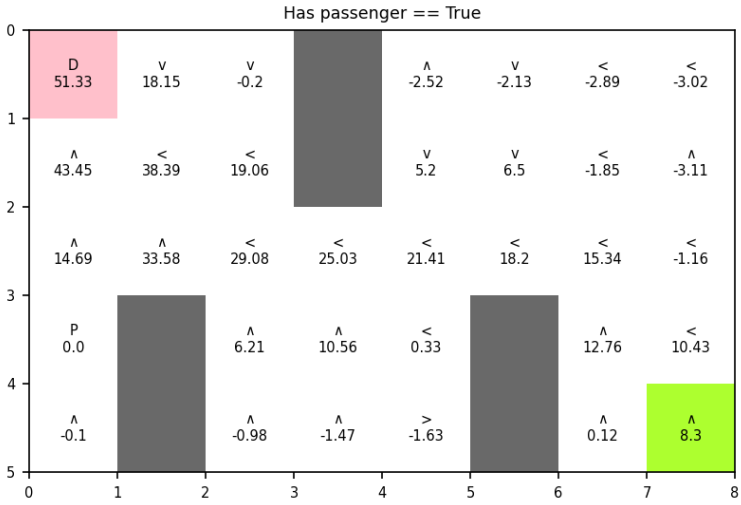
El agente inicia en posiciones aleatorias. Además de un Qvalor que depende de la posición y las acciones, se agregó el hecho de tener o no un pasajero en ese instante de tiempo como parte del estado.

4.1. $\epsilon = 0.1$

Con $\epsilon = 0.1$ la cantidad de iteraciones que demoró antes de converger fueron 212. En los estados más alejados cuando no tiene pasajeros no tiene una política óptima, en los que sí tiene pasajeros sí son más óptimas.



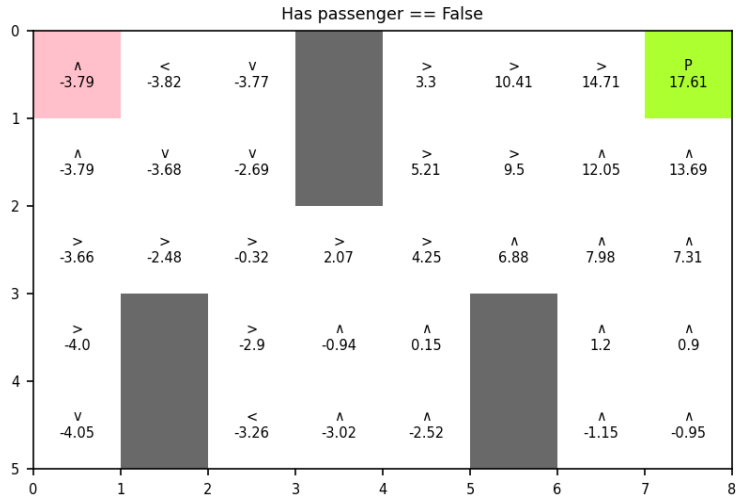
Taxi $\epsilon = 0.1$, *no tiene pasajero.*



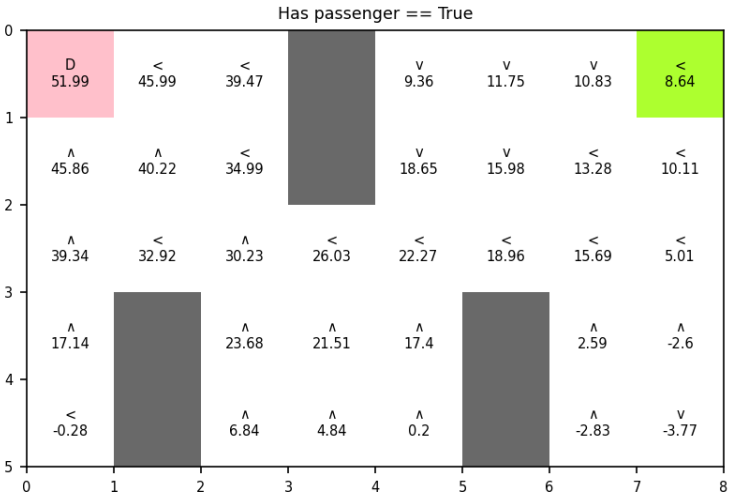
Taxi $\epsilon = 0.1$, *sí tiene pasajero.*

4.2. $\epsilon = 0.5$

Con $\epsilon = 0.5$ la cantidad de iteraciones que demoró antes de converger fueron 249. Nuevamente los estados más alejados de no tener pasajero no tuvieron política óptima. En los que sí tiene pasajero solo hubo 2 sin política óptima.



Taxi $\epsilon = 0.5$, *no tiene pasajero.*



Taxi $\epsilon = 0.5$, *sí tiene pasajero.*

4.3. $\epsilon = 0.9$

Correr el experimento para $\epsilon = 0.9$ se vuelve redundante para este punto. La tendencia es la misma que en los otros dos casos. Por error había puesto unos valores mucho más optimistas como recompensa para este experimento, y con las pruebas se volvió evidente que entre más valores óptimos se tenga, menos necesario es subir el epsilon a un valor cercano a 1. Con un epsilon de 0.5, una recompensa 4 veces mejor por dejar un pasajero y 5 veces menor por equivocarse, es suficiente para que todo sea una política óptima.