SARSA: State-Action-Reward-State-Action

En este ejercicio vamos a implementar el algoritmo de SARSA como un ejemplo de los métodos on-policy para el aprendizaje por refuerzo. Esto es, crear agentes que aprenden a alcanzar un objetivo específico.

El método de SARSA se basa en el cálculo de los q-valores utilizando los valores calculados para el estado de llegada siguiendo la fórmula de actualización de los q-valores:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s) + \gamma Q(s',a')]$$

Para implementar SARSA definiremos un agente, `sarsa_agent.py` el cual utilizaremos para interactuar con el ambiente de Gridworld.

Task 1

1. Implemente la classe `SARSA` con cinco atributos: - `epsilon` que corresponde a la estrategia de aprendizaje ϵ-greedy, `0.9` por defecto. - `gamma` que corresponde al factor de decuento a utilizar, `0.96` por defecto. - `alpha` que corresponde a la taza de aprendizajem `0.81` por defecto. - `Q` que almacena los q-valores del agente. - `env` que es una referencia al ambiente.

2. El comportamiento del agente (la interacción con el ambiente) esta dado por los métodos:

    - `choose_action` que recibe un estado como parámetro y retorna la acción a ejecutar para dicho estado siguiendo una estrategia ϵ-greedy.
    - `action_function` que recibe como parámetro los componentes de SARSA (estado1, acción1, recompensa, estado2, acción2) y calcula el q-valor `Q(estdo1, acción1)`.
3. La interacción entre el agente y el ambiente inicia desde el ambiente, que ejecuta cada interacción de SARSA para cada episodio. (1) La interacción comienza decidiendo la acción a tomar para el estado actual (la cual esta dada por el agente), (2) luego debemos ejecutar la acción, obteniendo el estado de llegada y la recompensa de ejecutar dicha acción, (3) luego calculamos la acción a tomar para el estado de llegada, (4) por último calculamos el q-valor definido por la función de las acciones.

Task 2

Implemente el ambiente de cliff-walk (basado en el ambiente de Gridworld utilizdo anteriormente) y resulevalo utilizando el método de SARSA. Recuerde que en este

ambiente la recompensa por caer al barranco es de -100 y la recompensa de cada paso es -1. Para la ejecución vamos a suponer acciones determinísticas.

Además responda las siguientes preguntas

1. ¿Cuál es el comportamiento del agente si utilizamos un factor de descuento de 1?
2. ¿Cómo podemos minimizar la trayectoria del agente entre el estado inicial y el estado de llegada?

Justifique sus respuestas con ejecuciones reales del agente.

In [17]:
```
from assignment_td_sarsa.environment_world import EnvironmentWorld, Action
from assignment_td_sarsa.sarsa_agent import SarsaAgent

cliff_world = EnvironmentWorld([
    ['-1'] * 12,
    ['-1'] * 12,
    ['-1'] * 12,
    ['-1'] + ['-100'] * 10 + ['1000']
],
    terminal_states=[(x, 3) for x in range(1, 12)],
    initial_state=(0, 3))
```

In [18]:
```
cliff_world
```

Out[18]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1C	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	1000

In [19]:
```
sarsa_agent = SarsaAgent(
    world=cliff_world,
    learning_rate=0.81,
    discount_factor=0.96,
    epsilon=0.9
)
```

In [20]:
```
sarsa_agent.iterate_learning(num_steps=1000000)
```

```
100%|██████████| 1000000/1000000 [00:12<00:00, 81760.00it/s]
```

In [21]:
```
sarsa_agent.print_policy()
```

```
      0      1      2      3      4      5      6      7      8      9   \
0   down  right   down   down  right   down   down   down  right  right
1  right  right  right  right  right  right   down  right  right   down
2  right  right  right  right  right  right  right  right  right  right
3     up   None   None   None   None   None   None   None   None   None

      10    11
0   down  down
1  right  down
2  right  down
3   None  None
```

In [22]:
```
sarsa_agent = SarsaAgent(
    world=cliff_world,
    learning_rate=0.81,
    discount_factor=1,
    epsilon=0.9
)
```

In [23]:
```
sarsa_agent.iterate_learning(num_steps=1000000)
```

```
100%|██████████| 1000000/1000000 [00:12<00:00, 80091.88it/s]
```

In [24]:
```
sarsa_agent.print_policy()
```

```
      0      1      2      3      4      5      6      7      8      9   \
0  right  right  right  right  right   down  right  right  right  right
1  right  right  right   down  right  right  right  right   down   down
2  right     up  right  right  right  right  right  right  right  right
3     up   None   None   None   None   None   None   None   None   None

      10    11
0   down  down
1   down  down
2  right  down
3   None  None
```

1. ¿Cuál es el comportamiento del agente si utilizamos un factor de descuento de 1?

En este caso no afecta la politica, ya que independientemente del descuento es mejor tomar el camino más corto, en parte esto se debe a que las acciones son deterministicas, si hubiera ruido en las acciones podríamos esperar cambios en la politica, porque sin el descuento podría valer más la pena evitar estar proximo al acantilado

2. ¿Cómo podemos minimizar la trayectoria del agente entre el estado inicial y el estado de llegada?

Como dije en el punto anterior al incrementar el descuento se crea un sesgo a menores trayectorias