Universidad ORT Uruguay

Facultad de Ingeniería

Inteligencia Artificial

Obligatorio

Entregado como requisito para la obtención del título de Ingeniero en Sistemas

Guzmán Dupont – 230263

Tutores: Federico Armando, Alejo Garat

2023

Etapa 1

Exploración

Comenzamos con los siguientes parámetros:

- 20 iteraciones de 100 partidas con parámetros:
    - epsilon: 1,0,
    - alpha: 0,1,
    - gamma: 0,9
- Los valores de ángulos y velocidades establecidos son:

```
#Angulos:
cost1_space = np.linspace(-1, 1, 10)
sint1_space = np.linspace(-1, 1, 10)
cost2_space = np.linspace(-1, 1, 20)
sint2_space = np.linspace(-1, 1, 20)
velt1_space = np.linspace(-12.57, 12.57, 40)
velt2_space = np.linspace(-28.27, 28.27, 30)
```

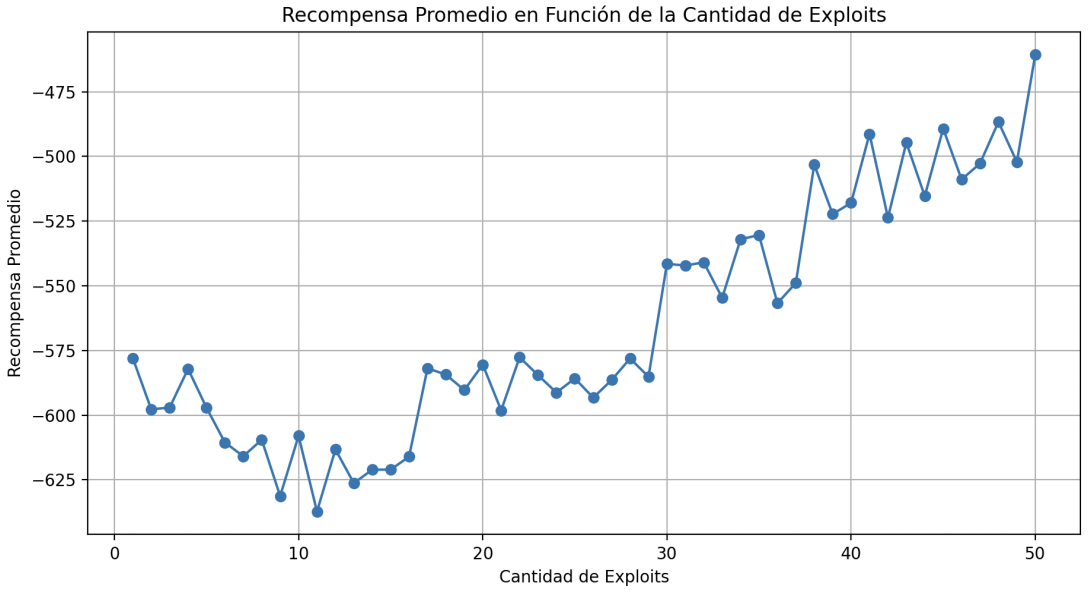
- El valor máximo de reward fue: -977,95
- El valor mínimo de reward fue: -699,25

Explotación

Ahora procedemos a hacer un exploit luego de que el agente haya experimentado con los valores previamente mencionados.

También procedemos a generar una gráfica que muestra el reward obtenido haciendo 50 exploits de 100 partidas.

Luego de las 50 iteraciones obtuvimos la siguiente gráfica:



Etapa 2

Exploración

Para esta etapa veremos de cambiar algunos parámetros con el fin de llegar a mejorar los resultados de la etapa anterior.

Definimos los siguientes parámetros:

- 20 iteraciones de 100 partidas con parámetros:
    - epsilon: 1,0,
    - alpha: 0,3,
    - gamma: 0,9.
- Los valores de ángulos y velocidades establecidos son:

```
#Angulos:
cost1_space = np.linspace(-1, 1, 15)
sint1_space = np.linspace(-1, 1, 15)
cost2_space = np.linspace(-1, 1, 25)
sint2_space = np.linspace(-1, 1, 25)
velt1_space = np.linspace(-12.57, 12.57, 45)
velt2_space = np.linspace(-28.27, 28.27, 35)
```

- Los resultados arrojados fueron los siguientes:
    - Mínimo reward: -684,44
    - Máximo reward: -696,8

No parece haber una mejoría significativa.

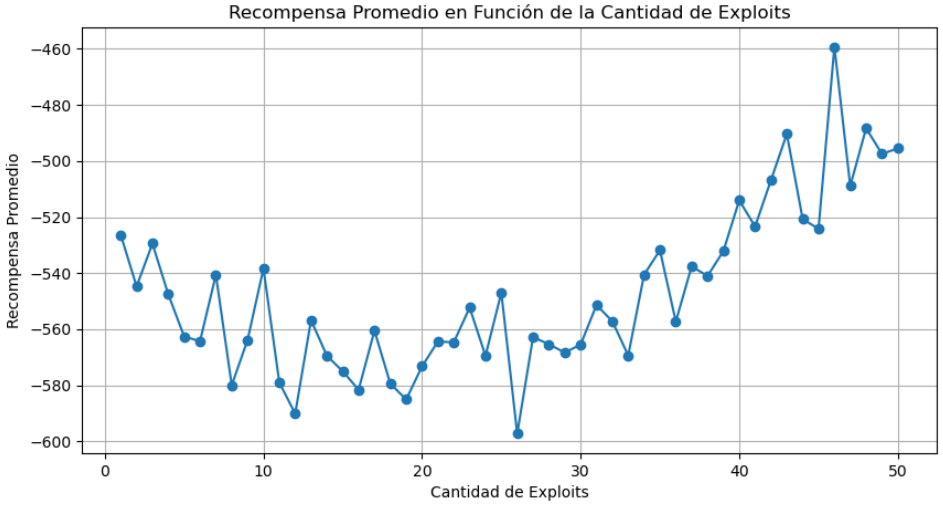
Aumentamos la cantidad de iteraciones a 40 para ver el resultado, mantenemos los demás parámetros intactos.

- Los resultados arrojados fueron los siguientes:
    - Mínimo reward: -679,7
    - Máximo reward: -697,47

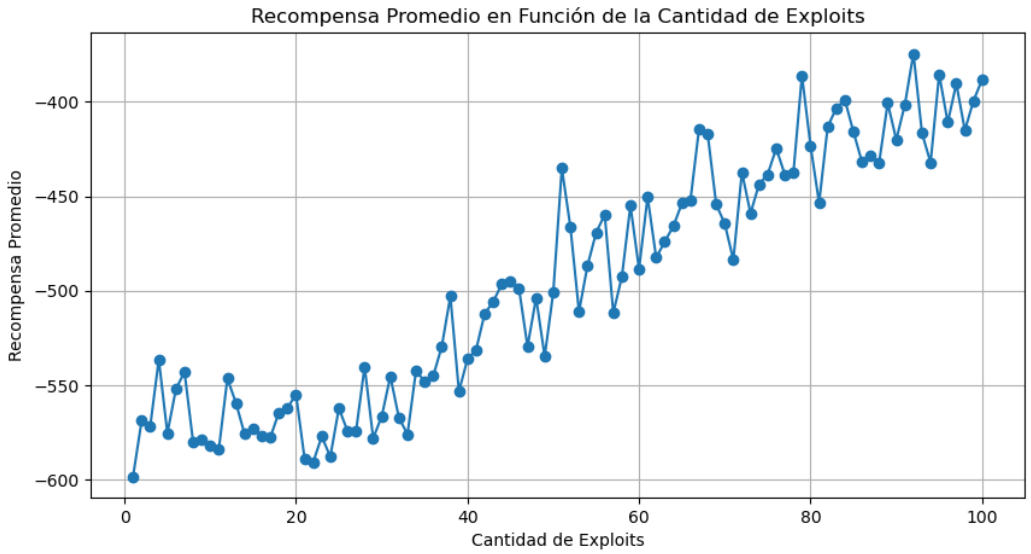
De todas maneras el promedio de estos 20 números nuevos fue de -691

Explotación

Luego de haber entrenado con los parámetros anteriores procedemos a hacer una explotación y compararla con la anterior. El número de iteraciones es 50 y la cantidad de partidas jugadas son 100.



La gráfica no muestra una mejora en los resultados de los exploits, si bien la mejor reward se dio en el -460 hacia el final de las iteraciones se estancaron los valores alrededor del -500 y nosotros buscamos mejorar eso.
Considero que darle más iteraciones a mi agente puede hacer que mejore los valores y el desempeño.
El número de iteraciones es 100 y el de partidas es 100.



Ahora podemos ver que el agente mejora la reward al aumentar el número de iteraciones. Esto nos permite avanzar a la siguiente etapa.

Etapa 3

Exploración

Para esta etapa volvemos a ajustar los parámetros del agente con el fin de mejorar el desempeño. Los parámetros son los siguientes:

- 40 iteraciones de 100 partidas con parámetros:
    - epsilon: 0,8,
    - alpha: 0,5,
    - gamma: 0,9.

Al cambiar a epsilon a 0,8 espero realizar un equilibrio entre la exploración (aproximadamente el 80% del tiempo) y la explotación (aproximadamente el 20% del tiempo).

- Los ángulos definidos son los siguientes:

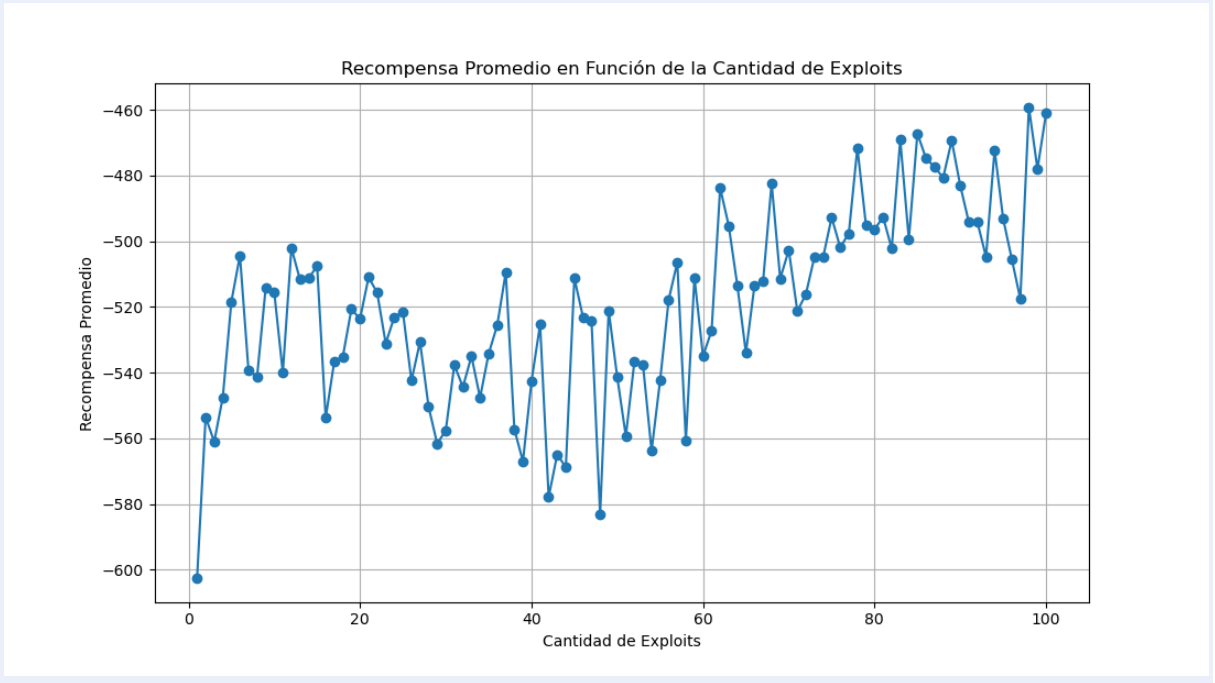
```
#Angulos:                                  Originalmente:
cost1_space = np.linspace(-1, 1, 20) # 5
sint1_space = np.linspace(-1, 1, 20) # 5
cost2_space = np.linspace(-1, 1, 30) # 10
sint2_space = np.linspace(-1, 1, 30) # 10
velt1_space = np.linspace(-12.57, 12.57, 50) # 30
velt2_space = np.linspace(-28.27, 28.27, 40) # 20
```

- Los resultados fueron los siguientes:
    - Reward mínimo: -662,77
    - Reward máximo: -692,74

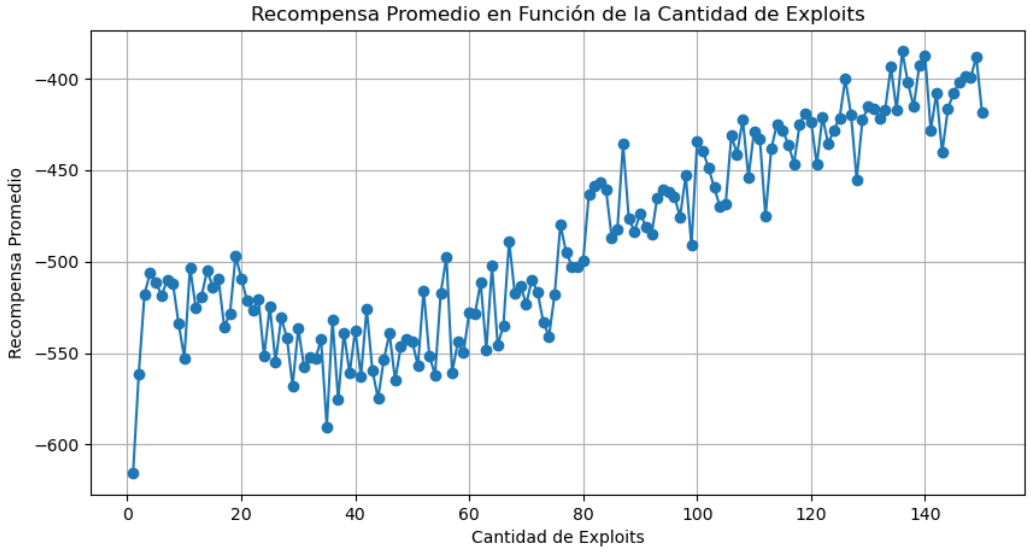
El promedio de las 20 primeras iteraciones fue de: -676.505
El promedio de las 20 iteraciones restantes fue de: -678.733
El promedio fue muy similar entre las iteraciones y mejor que las anteriores, esto nos brinda una sensación de tranquilidad porque el agente va mejorando.

Explotación

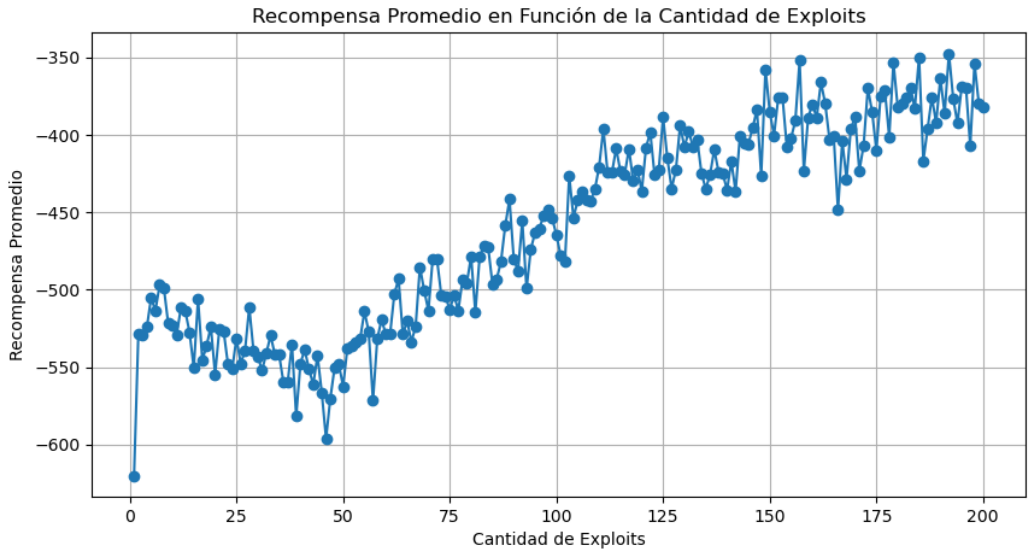
Generemos una exploración con los parámetros anteriores pero 100 iteraciones de 100 jugadas como la vez anterior y veamos cómo se comporta.



Estos resultados no son los esperados. Puede que falten iteraciones para que el agente aprenda, en la etapa anterior eso ayudó así que intentemos con eso. Hacemos 150 iteraciones.



Si bien mejoró, no es el resultado que estábamos buscando. En la etapa 2 se desempeñó de manera mucho mejor.
A modo de prueba se hizo una ejecución de 200 iteraciones y estos fueron los resultados.



Este resultado está mejor teniendo un zona donde la gráfica asciende, 50 hasta 110, y otra zona de la gráfica donde se mantiene bastante estable, desde 110 hasta 200. Sigamos con la siguiente etapa.

Etapa 4
Exploración
Para esta etapa volvemos a ajustar los parámetros del agente con el fin de mejorar el desempeño. Los parámetros son los siguientes:

- 40 iteraciones de 100 partidas con parámetros:
    - epsilon: 0,6,
    - alpha: 0,7,
    - gamma: 0,9.

- Mantendremos los ángulos como en la etapa anterior porque aumentar el intervalo de ángulos conlleva más tiempo de espera.

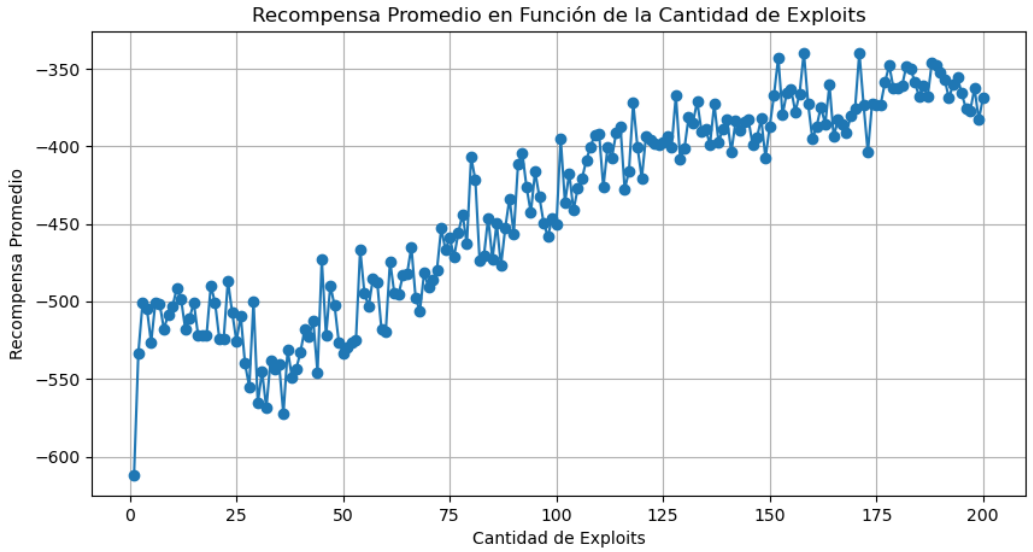
```
#Angulos:                                  Originalmente:
cost1_space = np.linspace(-1, 1, 20) # 5
sint1_space = np.linspace(-1, 1, 20) # 5
cost2_space = np.linspace(-1, 1, 30) # 10
sint2_space = np.linspace(-1, 1, 30) # 10
velt1_space = np.linspace(-12.57, 12.57, 50) # 30
velt2_space = np.linspace(-28.27, 28.27, 40) # 20
```

- Los resultados fueron los siguientes:
    - Reward mínimo: -634.03
    - Reward máximo: -669.28

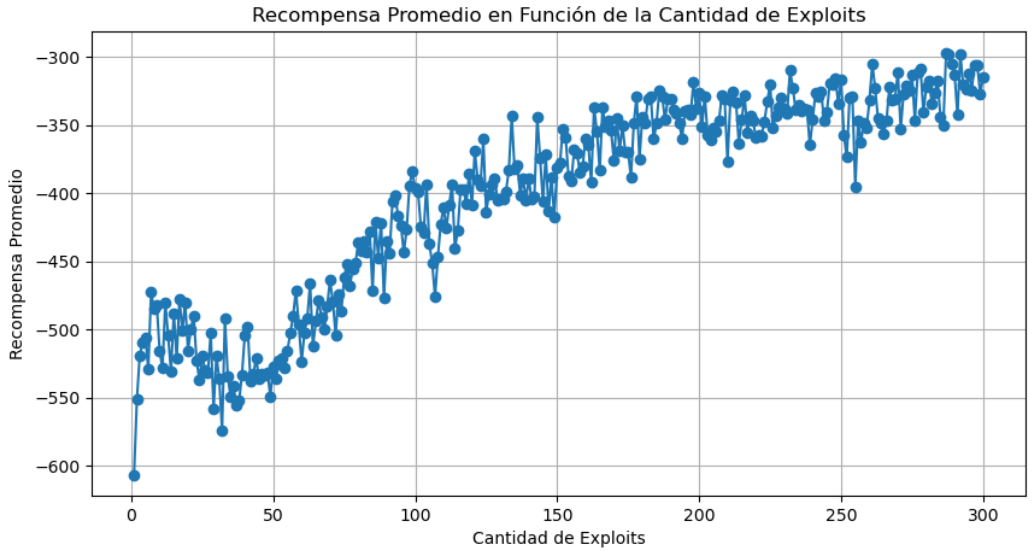
El promedio de las 20 primeras iteraciones fue de: -649.967
El promedio de las 20 iteraciones restantes fue de: -652.33

Al igual que en la etapa anterior el promedio de las primeras 20 iteraciones es menor al de las 20 restantes, interesante.

Explotación
En esta etapa de explotación haremos 200 iteraciones de 100 partidas con los parámetros establecidos anteriormente. Buscamos mejorar el reward de la etapa anterior.



Fue un resultado muy similar al anterior.
Decidimos aumentar la cantidad de iteraciones ya que en las etapas anteriores ha dado buenos resultados. Aumentamos la cantidad de iteraciones a 300 y este es el resultado:



Si bien la mejora no es tan significante, vemos que al aumentar la cantidad de iteraciones se llega a una reward mejor.

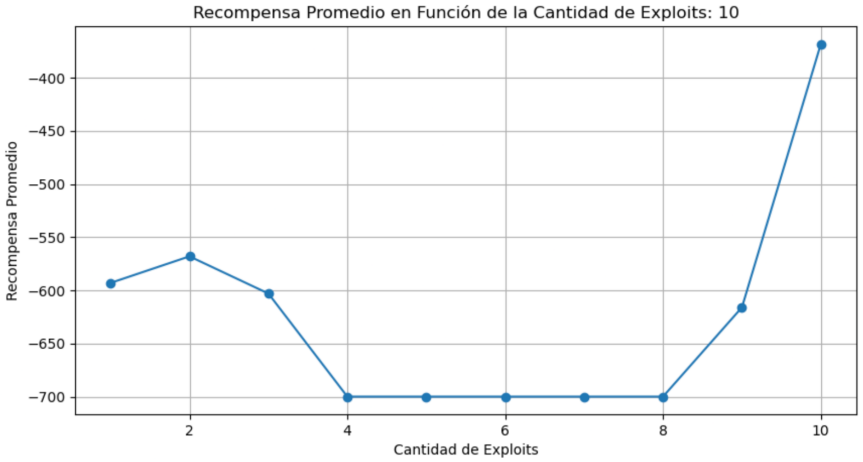
Etapa 5

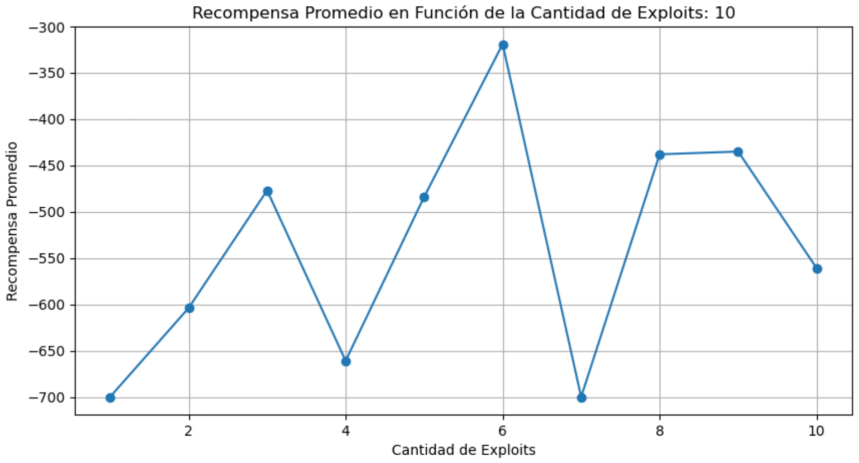
Por recomendación del profesor de práctico, para esta etapa se cambian algunos elementos que teníamos en las etapas anteriores.
Crearemos una función que explore y explote al mismo tiempo. Cambiaremos los valores de epsilon durante la etapa. También graficaremos la exploración para los distintos valores de epsilon.
La discretización se baja con el fin de reducir los tiempos de ejecución, pero se mantiene la cantidad de iteraciones.
Para esta parte de la etapa ajustamos los parámetros del agente con el fin de mejorar el desempeño. Los parámetros son los siguientes:

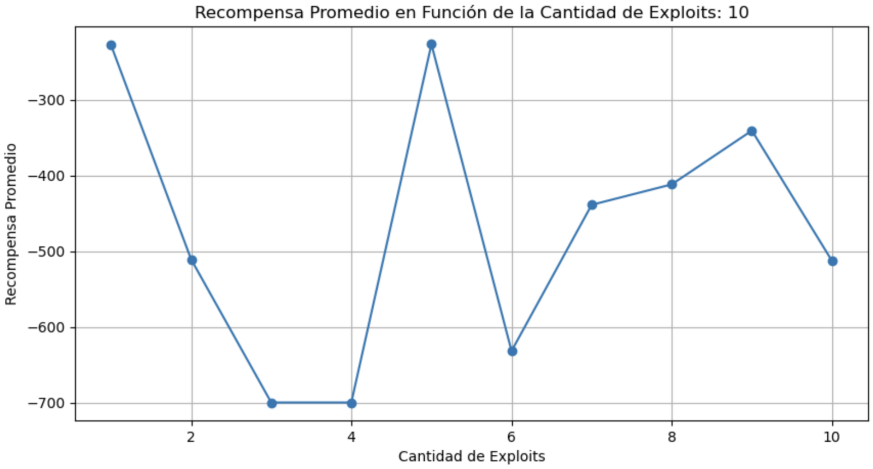
- 25 iteraciones de 25 partidas de exploración con parámetros:
    - epsilon: 0,9, inicialmente,
    - alpha: 0,7,
    - gamma: 0,9.
- Cambiamos los ángulos.

```
#Angulos:                                  Originalmente:
cost1_space = np.linspace(-1, 1, 10) # 5
sint1_space = np.linspace(-1, 1, 10) # 5
cost2_space = np.linspace(-1, 1, 20) # 10
sint2_space = np.linspace(-1, 1, 20) # 10
velt1_space = np.linspace(-12.57, 12.57, 40) # 30
velt2_space = np.linspace(-28.27, 28.27, 30) # 20
```

Cada dos iteraciones cambiamos el valor de epsilon, le restamos 0,02.
Al momento de explotar hacemos 10 iteraciones de 100 partidas tomando al maxQ y haciendo la gráfica de la reward de cada iteración.
Lo que no me termina de cuadrar es que los resultados de las gráficas parecen mostrar rewards independientes, es decir, parece que no aprende porque las recompensas son muy variables. Dejo algunas imágenes a manera de evidencia.







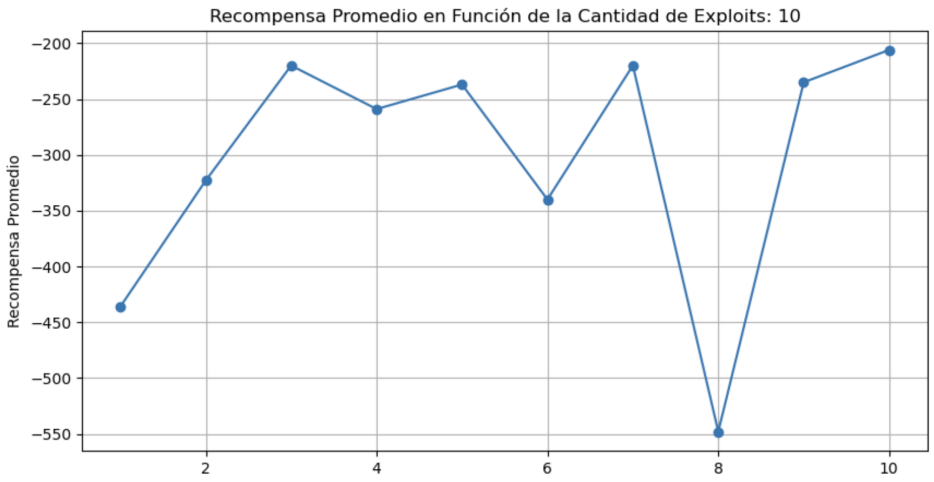
Las gráficas se fueron generando de manera secuencial a como están mostradas (la primera gráfica se generó primero, etc). Parece que las gráficas fueran de exploraciones independientes, sin embargo hubo un proceso de aprendizaje, en teoría, pero que no se refleja en los resultados.

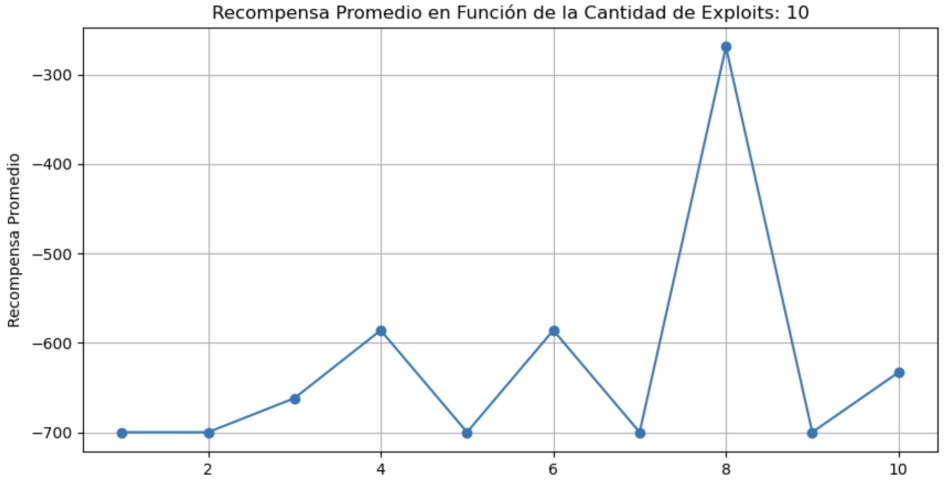
Dada la situación anterior, a modo de recomendación, se aumentó la cantidad de iteraciones de la función que explora y explota al mismo tiempo llamada *exec*.

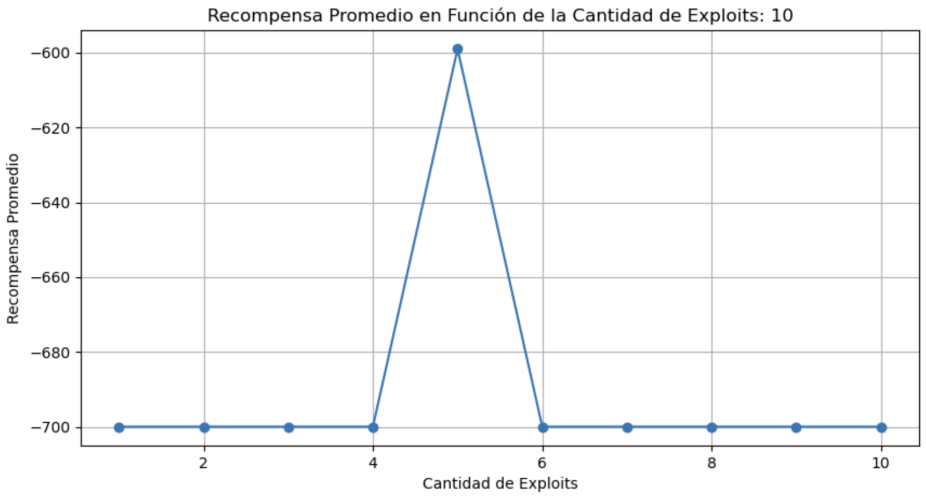
Sigo generando cambios en el código, aumentando la cantidad de iteraciones, pero el agente no mejora, todo lo contrario, empeora luego de 80 iteraciones de la función *exec* haciendo 25 iteraciones de exploración y 10 iteraciones[1] de exploits jugando 20 partidas.

Dejo la evidencia más abajo:

[1] La cantidad de iteraciones aumenta en 10 conforme aumenta las iteraciones de *exec*. Esto fue a modo de prueba con el fin de que el agente obtenga evaluaciones más robustas.







En la carpeta se encuentra el .pkl del Q con el que se obtuvieron las gráficas anteriores.