Taller 2: Aprendizaje por refuerzo aplicado a la solución del cubo de Rubik de 2x2x2 [1]

Autor Héctor Javier Rojas

***Abstract*— A comparison of the results of a proper implementation of a 2x2x2 Rubik's gym environment [1] is made using reinforcement learning as learning model.**

***Impact Statement* — se tratará de probar si es posible lograr una convergencia y que el modelo de aprendizaje pueda lograr buenos resultados en su función objetivo, se comparan diferentes estructuras de redes densas usadas.**

***Index Terms*— reinforcement learning, DQLearning [1], Red Densa, politicas, agentes inteligentes.**

I. Introduction

Se hace una implementación del gym de Rubik 2x2x2 para personalizar la función de recompensa que viene por defecto en el ambiente genérico, con esta modificación se hace una implementación de un modelo de aprendizaje por refuerzo QLearning usando un agente y una red densa para entrenar sus acciones

ambiente

Para iniciar se hizo una implementación propia del ambiente por defecto de RubiksCube222 [1], esto para poder sobre escribir la función de recompensa.

A. Función de recompensa [1]

La función que está escrita por defecto tiene únicamente dos posibles valores de recompensa, 100 para el caso en que el subo este completamente armado y -1 en caso contrario, de manera que el agente tiene problemas para lograr una ruta de acciones porque para el agente todas las rutas son iguales, mantienen el mismo premio -1.

B. Función de recompensa personalizada [1]

Para el caso personalizado se hace una asignación de recompensa por cada posición correcta en comparación al cubo ordenado, se hace una comparación individual de las 24 componentes individuales. Además, se asigna un bono si la una cara ha sido completada para esto se suma un valor de 10 por cada cara armada

II. PROCEDIMIENTO

una vez personalizada la función de recompensa, el modelo comienza a converger y la función de loss cae rápidamente a lo largo de las iteración, lo cual es un buen signo de ir en la dirección adecuada.

A partir de ahí se configuran los hiperparámetros correspondientes a la estructura de la red densa que entrenara las acciones, variando la cantidad de neuronas y cantidad de capas en la red densa y a continuación se presentaran los resultados obtenido para tres estructuras diferentes.

Naturalmente, las tres estructuras acá probadas difieren en tamaño así que entre más grandes son más tiempo toma entrenarlas, es un resultado que se esperaba obtener y que finalmente ocurrió, así que hacemos a un lado el resultado trivial y se va a comparar la convergencia y valor de la función objetivo.

A. *Red Neuronal Densa [1] - 300-150-100-50-20-3*

Se estructura una red neuronal de 6 capas con la siguiente topología.

- Capa 1 (Entrada) 300 Neuronas (*ReLu*)
- Capa 2 (Intermedia) 150 Neuronas (*ReLu*)
- Capa 3 (Intermedia) 100 Neuronas (*ReLu*)
- Capa 4 (Intermedia) 50 Neuronas (*ReLu*)
- Capa 5 (Intermedia) 20 Neuronas (*ReLu*)
- Capa 6 (Salida) 3 Neuronas (*ReLu*)

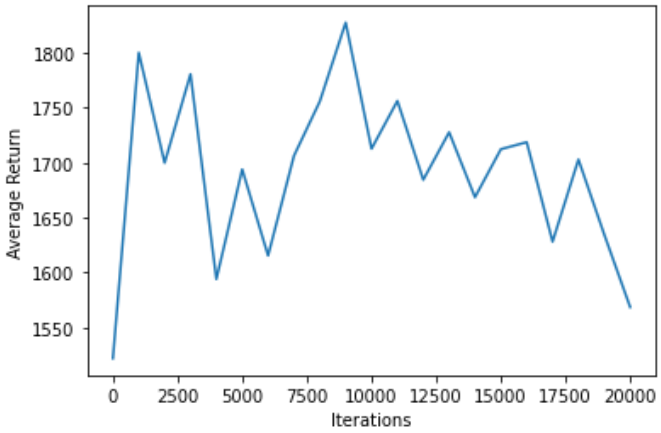

Fig 1- Curva de promedios de recompensas a lo largo de las iteraciones.

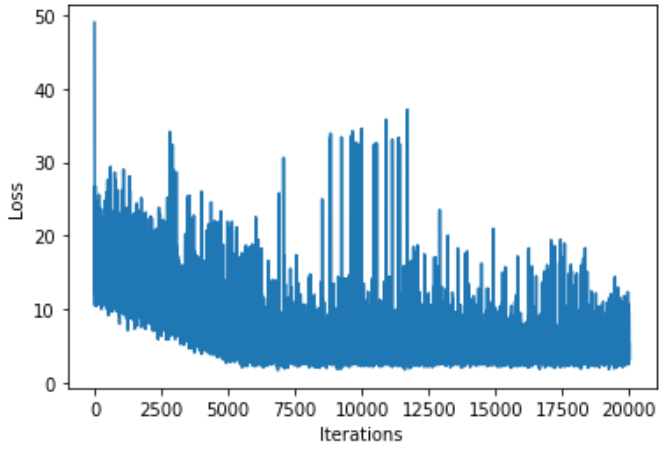

Fig 2 - Curva de función de perdida a lo largo de las iteraciones

B. *Red Neuronal Densa [1] - 5000 2000 1000 750 500 250 100 50 20 3*

Se estructura una red neuronal de 10 capas con la siguiente topología.

- Capa 1 (Entrada) 1000 Neuronas (*ReLu*)
- Capa 2 (Intermedia) 750 Neuronas (*ReLu*)
- Capa 3 (Intermedia) 500 Neuronas (*ReLu*)
- Capa 4 (Intermedia) 250 Neuronas (*ReLu*)
- Capa 5 (Intermedia) 100 Neuronas (*ReLu*)
- Capa 6 (Intermedia) 50 Neuronas (*ReLu*)
- Capa 7 (Intermedia) 20 Neuronas (*ReLu*)
- Capa 8 (Salida) 3 Neuronas (*ReLu*)

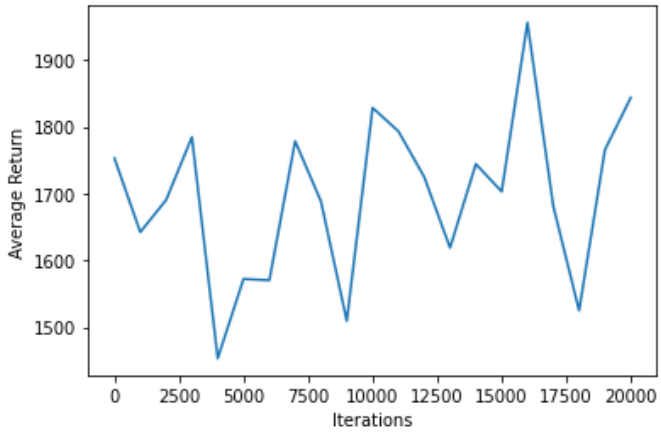

Fig 3-Curva de promedios de recompensas a lo largo de las iteraciones.

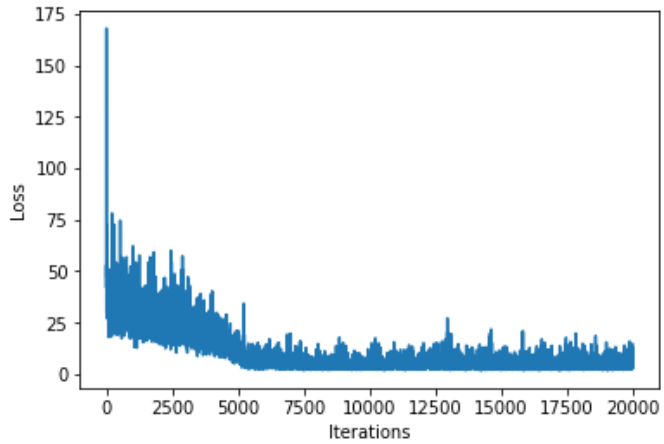

Fig 4 - Curva de función de perdida a lo largo de las iteraciones

C. *Red Neuronal Densa [1] - 5000 2000 1000 750 500 250 100 50 20 3*

Se estructura una red neuronal de 10 capas con la siguiente topología.

- Capa 1 (Entrada) 5000 Neuronas (*ReLu*)
- Capa 2 (Intermedia) 2000 Neuronas (*ReLu*)
- Capa 3 (Intermedia) 1000 Neuronas (*ReLu*)
- Capa 4 (Intermedia) 750 Neuronas (*ReLu*)
- Capa 5 (Intermedia) 500 Neuronas (*ReLu*)
- Capa 6 (Intermedia) 250 Neuronas (*ReLu*)
- Capa 7 (Intermedia) 100 Neuronas (*ReLu*)
- Capa 8 (Intermedia) 50 Neuronas (*ReLu*)
- Capa 9 (Intermedia) 20 Neuronas (*ReLu*)
- Capa 10 (Salida) 3 Neuronas (*ReLu*)

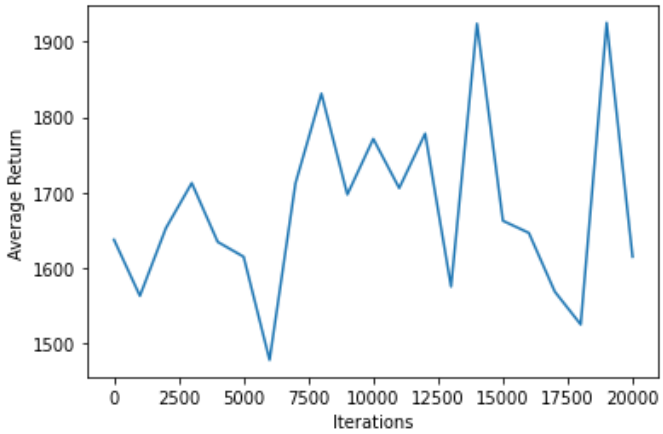

Fig 5 - Curva de promedios de recompensas a lo largo de las iteraciones.

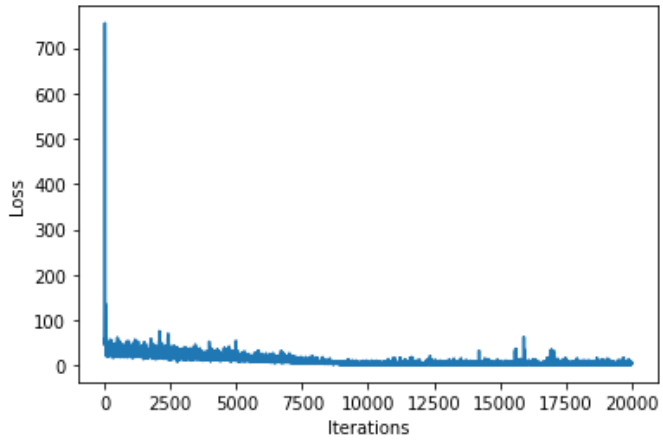

Fig 6 - Curva de función de perdida a lo largo de las iteraciones

III. Observaciones

A. *Entrenamientos*

Queda en evidencia que no aparentemente no importa la estructura ni tamaño de la red neuronal, los resultados promedios parecen estar oscilando alrededor de los mismos valores entre 1500 y 1900 [1] [2], sin embargo, el agente no logra hacerlos subir de ahí

Por otro lado, las funciones de perdida de todos los entrenamientos parecen converger rápidamente a pesar de que en cada iteración la perdida parece incrementarse súbitamente peor tiene a converger a partir de ahí

Se experimento también con los otros hiperparámetros [1] de ejecución sin embargo no tuvieron impactos significativos en los resultados del entrenamiento todo permaneció relativamente constante a excepción de los tiempos de entrenamiento que si se vieron incrementados a mayor tamaño de la red o un incremento de las interacciones

IV. Conclusiones

- Con la configuración actual a pesar de que la red converge, no logra atravesar cierto estado y parece estar oscilando sobre un grupo de ellos
- Con las condiciones actuales parece no tener importancia la topología de la red dentro del comportamiento del entrenamiento
- Para trabajos posteriores se recomienda hacer cambios en las políticas del agente para hacerlo más dinámico y no quede atrapado en mínimos locales
- No se logra la solución del cubo de Rubik de 2x2x2

V. Bibliografía

[1] K. a. K. Hukmani y S. a. V. Sreekumar, «RubiksCubeGym,» GutHub, 2021. [En línea]. Available: https://github.com/DoubleGremlin181/RubiksCubeGym. [Último acceso: 27 05 2022].

[2] «Google Colab,» 27 05 2022. [En línea]. Available: https://www.tensorflow.org/agents/tutorials/1_dqn_tutorial#metrics_and_evaluation.

[3] V. Mnih1, K. Kavukcuoglu1, D. Silver1, A. A. Rusu1, J. Veness1, M. G. Bellemare1, A. Graves1 y M. Riedmiller1, «Human-level control through deep reinforcement,» *Nature Vol 518*, pp. 531-533, 2015.