Memoria Machine Learning

El objetivo de este proyecto era aprender como se puede implementar un algoritmo de reinforcement learning a un juego simple. Empecé intentando hacerlo con el del dinosaurio, pero ajustar el sistema de recompensas de la red neuronal es algo complicado, y entrenar el modelo no es tan simple como hacer una red neuronal de unas cuantas capas y dejar que corra.

Al final he optado por hacer un curso en Kaggle de Reinforcement Learning y usar eso para realizar el proyecto. En este proyecto se utiliza Reinforcement Learning para jugar al Conecta 4, un juego bastante simple ya que está basado en turnos y hay una posibilidad finita de tableros (aunque sean muchos tableros).

Empecé el proyecto con algo bastante simple, viendo cómo funcionaba la implementación de Kaggle-environments del Conecta 4 y conociendo las funciones que tiene. El primer agente que hice fue uno muy simple que simplemente pone fichas o de manera aleatoria o en una columna predeterminada. Estos algoritmos no llegaron muy lejos, que se diga.

Acto seguido, siguiendo con el curso, hice un modelo heurístico. Para esto use varias funciones proporcionadas por kaggle, que sirven para analizar la tabla. Con este algoritmo se vio una mejora notable respecto al anterior, ya que este juega reaccionando a los movimientos del oponente e intentando predecir el próximo movimiento. Aún así, al solo predecir un movimiento, había muchas situaciones en las que realiza un movimiento que consideraba bueno que resultaba ser perjudicial.

Esto se puede mejorar aumentando la profundidad de la predicción e implementando algún algoritmo de decisión. En la competición de Kaggle utilizan uno llamado minimax, pero este es computacionalmente muy pesado cuando la profundidad del árbol de juego es alta. Esto se puede reducir usando poda alfa-beta, que es el algoritmo implementado en el proyecto.

Por último, se utiliza Reinforcement Learning para acabar el proyecto. Para esto, el paquete tradicional de tensorflow se puede utilizar, pero hay paquetes como stable\_baselines y Gym de OpenAI que hacen la vida bastante más fácil a la hora de implementar modelos de RL. Gym nos permite adaptar el juego para que acepte recompensas, de manera que se pueda aplicar RL, y Stable\_Baselines nos ofrece varios modelos de RL que se pueden utilizar.

Esta parte es sin duda la más compleja, ya que los tiempos de entrenamiento ya empiecen a ser altos (~1 hora), y entender lo que está pasando es ya bastante complicado.

Una vez definida la clase con gym, se tiene que utilizar una red convolucional (simple, de dos capas) con una flatten y un output. A diferencia de otras redes neuronales que hemos visto en clase, esta tiene una capa de 512 porque no realiza una decisión, sino que va a actuar de función heurística para evaluar la efectividad de cada movimiento.

Una vez definida la red neuronal hay que utilizar los algoritmos de aprendizaje. En este caso se emplean tres: PPO1, A2C y ACER. Stable\_baselines hace esto bastante simple, y nos permite crear un modelo para jugar en dos líneas de código. Aún así, entender los detalles de cada modelo es bastante complicado, y no es algo que haya conseguido en este proyecto.

En general ha sido un proyecto bastante interesante, y un buen comienzo en el campo de RL.