Aprendizaje por Refuerzo - Diez Mil

Inteligencia Artificial y Neurociencias

Bonás, Valentin Giorgi, Federico Levi, Chantal Loza Montaña, Gastón

Descripción del juego

En este proyecto, aspiramos a crear y entrenar un agente que, mediante aprendizaje por refuerzo, aprenda a jugar al juego de dados llamado Diez Mil, con el objetivo de minimizar la cantidad de turnos que le tome ganar. El algorítmo utilizado es Q-learning tabular, siguiendo una política ϵ-greedy a la hora de entrenar.

Diez Mil es un sencillo juego de dados, donde los jugadores tiran dados con el objetivo de sumar puntos. La manera de sumar puntos se puede ver en la siguiente tabla:

Combinacion de dados	Puntos
1	100 puntos
5	50 puntos
1, 1, 1	1000 puntos
#, #, #	# * 100 puntos
1, 2, 3, 4, 5, 6	3000 puntos
3 pares	1500 puntos
6 iguales	10000 puntos

Una vez lanzados los dados, el jugador puede elegir cuales dados utiliza para sumar puntos y cuales para seguir tirando e intentando alcanzar mas puntos. Esto quiere decir que, por ejemplo, si el jugador lanza "1, 1, 1, 2, 4, 6", podría elegir sumar 1000 puntos (1,1,1) y tirar los otros tres dados restantes para intentar seguir sumando, podría elegir tirar todos los dados de nuevo o podría elegir sumar 1000 puntos y no volver a tirar con los otros tres dados restantes. Esta última decisión toma sentido cuando vemos la regla de que, si tiramos dados y no obtenemos ninguna combinación que de puntos, no solo no sumamos nada si no que perdemos lo sumado en el turno y nuestro turno termina automáticamente.

A fines de simplificar el entrenamiento, cambiaremos levemente las reglas del juego y diremos que si en nuestra tirada hay combinaciones que suman puntos, todas deben ser utilizadas (ej. Si tiramos "1,1,1,5,2,6" si o si debemos sumar los puntos de los tres unos y del 5, no podemos por ejemplo elegir volver a tirar el 5 en búsqueda de sumar mas puntos). Esto puede interpretarse también como una política hardcodeada, nuestro agente siempre utiliza los dados para sumar puntos en caso de ser posible.

Definicion del Ambiente y Estados

El ambiente es el entorno con el que el agente interactúa y devuelve ciertos outputs dependiendo de las acciones del agente. Los puntos claves del nuestro se encuentran a continuación.

1. Posibles acciones

Un agente, para un estado específico puede decidir entre tomar únicamente dos acciones, seguir tirando (la llamaremos "Jugar") o recolectar los puntos del turno y pasar al siguiente (la llamaremos "Plantarse").

Pero, ¿qué definimos como estado en este contexto?

2. Estados

Un estado es la posible configuración del ambiente en un momento dado. La información que guardemos para describir cada estado, es decir, cada posible configuración del ambiente, impacta fuertemente en el aprendizaje del

agente, ya que este aprende que acción es la indicada en cada estado. En nuestro caso, decidimos definir a un estado con dos atributos, la cantidad de dados disponibles para tirar y los puntos acumulados del turno, que empiezan en cada turno siendo seis dados y cero puntos. Hay otros dos factores que tienen buenos argumentos para ser guardados ya que otorgan informacion valiosa para decidir como actuar en un momento dado:

1. Exactamente que valores tomaron los dados que lanzamos.
2. Cantidad de puntos totales de la partida.

Decidimos no guardarnos exactamente los valores ya que al simplificar el juego diciendo que siempre que pueda hacer puntos con los dados los sumo, no nos importa que tiramos si no los puntos que nos dió y con cuantos dados nos quedamos para realizar nuestra proxima acción. Por otra parte, la cantidad de puntos totales parece información mas relevante, pero si tomamos en cuenta que hay aproximadamente 400 valores posibles para ellos (de 0 a 20000, yendo de 50 en 50), sumado a los casi 400 posibles valores para lo sumado en un turno (mismo cálculo, la lógica detrás de ambos para no ser 10000 es que podría estar en 9950 y sumar 10000 en una tirada) y los 7 posibles valores de la cantidad de dados a tirar (de 0 a 6), la cantidad de estados totales se iría a aproximádamente $400 \cdot 400 \cdot 7 = 1120000$ estados, haciendo el entrenamiento mucho mas complicado para un approach de Q-Learning tabular, que es el algorítmo que buscamos implementar.

Consideramos mas importante la cantidad de dados, ya que determina las probabilidades de lograr sumar puntos o no y los puntos del turno ya que nuestra acción de seguir jugando puede implicar perderlos (y intuitivamente nos parece positivo que nuestro agente entienda cuanto está arriesgando al momento de elegir dicha acción).

3. Sistema de recompensas

Para las recompensas probamos distintos approaches:

1. Que la recompensa sea 0 en todos los turnos y cuando termina el juego restarle la cantidad de turnos (sin descontar).
   - La idea detrás es que nuestro objetivo es minimizar la cantidad de turnos, entonces dar una recompensa que esté directamente relacionada parece ser una buena opción. Probamos tanto con inicializaciones negativas de los estados y con inicializacion = 0, pero no llegamos a resultados muy buenos, era prácticamente igual que el agente que siempre se planta.
2. Poner un objetivo de ser menor a X cantidad de turnos, sumar 1 si lo logra, 0 si es X y -1 si no (sin descontar).
   - Intentamos imitar la estructura vista en BlackJack, pero no fue nada util, siendo igual o incluso peor que el agente que siempre se planta.
3. Que se sume el score total del turno en cada acción, y si lo pierde que esa accion lo reste.
   - Con este approach obtuvimos buenos resultados preliminares, y agregando cosas como dividir el score negativo por la cantidad de dados a tirar (si pierde tirando 6 dados no queremos penalizar tanto porque tuvo muy mala suerte) y agregar una recompensa final al terminar el juego de 10000 puntos (descontada a medida que pasan las acciones hacen que el agente quiera terminar lo antes posible) conseguimos resultados ya si bastante mejores al agente que siempre se planta y al random, por lo que decidimos seguir con este setup a la búsqueda de hiperparámetros.

Búsqueda de hiperparámetros

Con el setup ya armado con las decisiones explicadas anteriormente, decidimos realizar una breve búsqueda de hiperparámetros con un grid search que incluye valores en cada posible variable los cuales nos parecían razonables y con los que probando un poco manualmente obtuvimos mejores resultados.

Los resultados fueron un alpha (α) de 0.05 gamma (γ) de 0.7 y un epsilon (ϵ) de 0.05.

Resultados

Tras realizar un entrenamiento de 1000000 episodios y hacer jugar a 50 agentes 10000 partidas cada uno, obtuvimos un promedio de 21.7 turnos para finalizar el juego, superando ampliamente al agente random y al que siempre decide plantarse, los cuales realizando el mismo testeo nos dieron promedios de 25.3 y 27.2.

La política aprendida por nuestro agente se puede encontrar en `agent_policy.json`.