Trabajo Práctico 1 - Aprendizaje por Refuerzo

Inteligencia Artificial y Neurociencias

Segundo semestre de 2024

El objetivo de este trabajo práctico es construir un agente capaz de jugar al juego de dados "Diez Mil".

Las **reglas del juego** están explicadas en detalle en `https://crearjuegos.ar/imprimi-y-juga/ver/10.000`.

Jugaremos una variante con un único jugador, considerando solo las siguientes formas de sumar puntos:

Dados	Puntaje	
1	100 puntos	
5	50 puntos	
1 1 1	1.000 puntos	
# # #	# × 100 puntos	(ejemplos: 2 2 2 = 200 puntos; 6 6 6 = 600 puntos)
1 2 3 4 5 6	3.000 puntos	(escalera)
3 pares	1.500 puntos	(incluyendo 4 iguales y un par)
6 iguales	10.000 puntos	(termina el juego y gana)

Notar que en esta variante, los puntos siempre se consiguen con los nuevos dados, y no hay combinación o acumulación posible con los dados ya tirados.

Además, no tendremos en cuenta la regla de los 750 puntos necesarios para "entrar al juego". La primera anotación se puede hacer con cualquier cantidad de puntos.

Se pide implementar un agente de aprendizaje por refuerzo y un ambiente en el cual el agente puede entrenarse. Esto incluye definir la noción de estado del ambiente, que puede no coincidir con el estado del juego.

Se proveen los siguientes archivos:

- `diezmil.py`: Clase `JuegoDiezMil`, que permite jugar un juego completo a un `Jugador`.
- `jugador.py`: Clase abstracta `Jugador` y dos implementaciones de muestra: `JugadorAleatorio`, que elige sus jugadas al azar, y `JugadorSiempreSePlanta`, que siempre elige plantarse.
- `utils.py`: Funciones y constantes útiles a todo el proyecto. Por ejemplo, `puntaje_y_no_usados(dados)` computa los puntajes de una tirada de dados.
- `template.py`: Tiene una propuesta de clases y métodos que podrían implementar como solución. Es simplemente una guía, no es necesario usarla o completarla.

- `entrenar.py`: Script para entrenar y almacenar la política de un agente siguiendo la interfaz presentada en `template.py`.
- `jugar.py`: Script que carga un jugador entrenado y ejecuta un juego completo de Diez Mil.

Este TP debe realizarse en grupos de **dos (2) a cuatro (4) integrantes**. Será puntuado con una nota binaria, de 0 puntos, o bien de 100 puntos. **La entrega no es obligatoria**; quienes no entreguen tendrán nota 0. (Recomendamos revisar las reglas de aprobación de la materia.)

Para lograr 100 puntos en este TP, se deberá:

1. Entrenar un agente con alguno de los algoritmos de RL vistos en clase (p.ej., Monte Carlo o Q-Learning, en alguna de sus variantes).
2. Con la política aprendida por ese agente, implementar y entregar un `JugadorEntrenado` que se ejecute correctamente en la clase `JuegoDiezMil`.
3. Presentar todos los archivos de código, junto a un breve informe (máximo dos carillas) detallando las decisiones tomadas, el grado de éxito que tuvieron y cualquier otro comentario que consideren relevante para entender el trabajo realizado.

Fecha límite de entrega: jueves 5/9 a las 23:59hs. Este TP no tiene recuperatorio.

Comentarios adicionales:

- La mayor dificultad del problema pasa por el diseño del ambiente: definir los estados, las acciones, las recompensas, etc. Eso, junto con el ajuste de los distintos parámetros (por ejemplo, ϵ, γ, α), es mucho más desafiante (y más interesante) que la escritura de los algoritmos de RL en sí misma.
- **Sugerencia:** Para simplificar el problema, sugerimos permitir al agente hacer solo dos acciones posibles: plantarse, o tirar todos los dados que no suman puntos. Es decir, excluir la posibilidad de tirar un subconjunto de dados.
- **Opcional para entusiastas:** En una implementación de Q-Learning, reemplazar la tabla Q por un perceptrón multicapa, para así transformarlo en Q-Learning no tabular.
- **Torneo:** Con todos los agentes entregados, haremos un campeonato para elegir a los mejores jugadores de Diez Mil!
- Si encuentran bugs en nuestro código, por favor avisar.