Corte Supremo

Teoría de Juegos

7 de junio de 2023

Integrantes

- Manuel Panichelli, L.U. 72/18
- Ignacio Alonso Rehor, L.U. 195/18

Introducción

En este trabajo implementamos un programa que simula el proceso de *Cut & Choose* iterado con información completa. Dos jugadores deben repartir entre ellos una torta. El primero hace el corte (*cut*) y el segundo elije la parte que más le conviene (*choose*).

En este informe contamos el trabajo que hicimos pero no cómo correrlo. Para ver instrucciones sobre eso consultar el `README.md`

Definiciones

La torta es "unidimensional" y está compuesta por 4 componentes comestibles (1, 2, 3, 4). Por ejemplo, una torta de tamaño 10 puede ser

$$1413234121.$$

Además, para cada jugador se tiene una **tabla de gustos** que indica cuanto le gusta cada componente: f_1 y f_2 para el jugador 1 y 2 respectivamente. En el trabajo probamos diferentes clases de tablas y vemos cómo esto altera los resultados.

Definimos la **utilidad** u_i para cada jugador $i = 1, 2$ en proporción a la cantidad de cada componente que hay en la torta. Sea una torta de tamaño N con partes $p_1, \dots p_N$

$$u_i(p_1 \dots p_N) = \frac{1}{\alpha} \sum_{j=1,\dots,N} f_i(p_j)$$

donde α es un factor que normaliza u_i a 1, que lo calculamos como la utilidad de la torta entera.

Mecánica del juego

En cada iteración se genera una nueva torta, y luego el jugador 1, bajo algún criterio que explicaremos a continuación, divide la torta con un solo corte (*cut*), mientras que el segundo elige una de las dos mitades (*choose*), quedandole así la otra al primer jugador. Iremos sumando las ganancias dadas por u_i a lo largo de cada iteración.

Objetivo

La idea del trabajo es explorar la diferencia de ganancias entre los diferentes criterios de elección de corte para el jugador 1. Estos consisten en el conocimiento y no de la función de utilidad del jugador 2. Al conocerla, intentaremos dar una estrategia para que el jugador 1 obtenga la mayor ganancia posible.

Implementación

Generación de tortas aleatorias

La generación aleatoria de una torta de tamaño N consiste en realizar N elecciones sobre el conjunto de gustos. Esto nos da una torta cuyos gustos no están necesariamente agrupados, y cuya elección resultó igual de probable que el resto de estos.

Para facilitar la implementación decidimos usar la función `random.choices` de la biblioteca estándar de Python.

Tipos de información del jugador 1 (Criterios de elección de cortes)

Implementamos dos tipos de información para el jugador 1, **información completa** e **incompleta**, en donde varía el conocimiento del jugador 1 sobre la función de utilidad del jugador 2 (u_2). Esto va a resultar interesante para cuantificar qué tan útil es contar con esa información.

Las implementamos de la siguiente forma:

- **Información incompleta**: como no conoce u_2, tiene que hacer un corte tal que sin importar que parte elija el 2 a él le de igual.

  Para ello, buscamos el corte que minimice la diferencia en la utilidad de las dos partes. Más formalmente, dada una torta $p_1 \dots p_N$ buscamos un corte c tal que

$$\arg\min_{c\in\{1,\dots,N\}} |u_1(p_1 \dots p_c) - u_1(p_{c+1} \dots p_N)|$$

- **Información completa**: En este caso, dado que el jugador 1 conoce la función de utilidad del jugador 2 (u_2), la estrategia va a consistir en buscar el corte c tal que maximice u_1 de la parte de la torta que no será la elección del jugador 2. Esto es,

  $$\arg\max_{c\in\{1,\dots,N\}} v(c),$$

  donde

  $$v(c) = \begin{cases} u_1(p_1, \dots, p_c) & \text{si } u_2(p_1, \dots, p_c) < u_2(p_{c+1}, \dots, p_N) \\ u_1(p_{c+1}, \dots, p_N) & \text{si no} \end{cases}$$

Tablas de valoración

Pensamos en 2 clases de equivalencia de tablas de valoración: **valoraciones iguales** y **valoraciones distintas**. Implementamos 4 pares de tablas concretas, que nombramos para poder referirnos a ellas en los gráficos

- **Identidad**

  Ambos tienen los mismos gustos, que son iguales al número asignado a cada componente

Gusto	f_1	f_2
1	1	1
2	2	2
3	3	3
4	4	4

- **Opuestos**

  Como la identidad, pero con gustos opuestos

Gusto	f_1	f_2
1	1	4
2	2	3
3	3	2
4	4	1

- **Opuestos disjuntos**

Tienen gustos opuestos, pero no comparten ninguno. Hay gustos que le gustan a 2 y no a 1 y viceversa.

Gusto	f_1	f_2
1	1	0
2	1	0
3	0	1
4	0	1

- **Igual vs picky eater**

  Este es diferente al resto. Al jugador 1 le dan igual todos los gustos, mientras que al jugador 2 solo le gusta el 1 (es un *picky eater*).

Gusto	f_1	f_2
1	1	1
2	1	0
3	1	0
4	1	0

Resultados

Corrimos la simulación para todas las combinaciones de información del jugador 1 (`completa` e `incompleta`) y tablas de valoración (`identidad`, `opuesto`, `opuesto_disjunto` e `igual_vs_picky`). Para cada una, usamos tortas de longitud 100 y corrimos 1000 iteraciones.

Primero observemos la diferencia entre **tipos de información**.

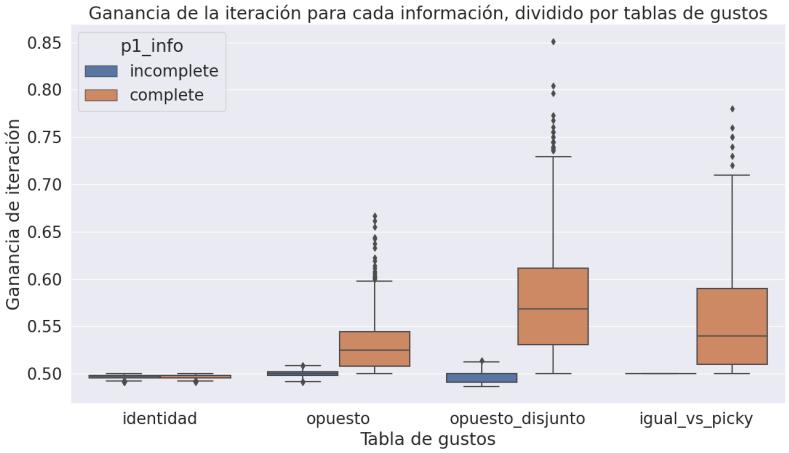

Figura 1: Ganancia de la iteración para cada información, dividido por tabla de gustos

en la figura 1 se ve para cada valuación la distribución de las ganancias que tuvo cada tipo de información del jugador 1 en cada iteración (*sin tomar en cuenta al jugador 2*). Esto nos permite ver en general que ganancia se puede asegurar un método en cada iteración del juego.

- Información incompleta siempre tiene una ganancia que ronda 0.5, pero no puede garantizárselo siempre como en la formulación teórica del método. Esto sucede porque la torta es discreta y en general cada gusto tiene un valor diferente. Por lo tanto, no siempre existe un punto en el que la valuación de cada mitad es exactamente 0.5.
- Para la identidad no fue mejor tener información completa, ya que como tienen la misma valuación todo lo que él valora lo valora de igual manera el jugador 2, no hay forma de sacar ventaja dándole una mitad menos valiosa (ya que valen lo mismo)
- En cambio, para el resto si pudo sacar ventaja. Al no ser iguales las tablas de gustos, el jugador 1 puede cortar la torta de forma tal que ambos se vean beneficiados, así logrando que el jugador 2 elija una mitad que lo beneficia más a él.
- Un resultado interesante es la diferencia en las distribuciones de ganancias entre las tablas **opuesto** y **opuesto_disjunto** bajo el supuesto de información completa. Notemos que, al no solaparse los gustos preferidos en la tabla de gustos **opuesto_disjunto**, el jugador 1 puede incorporar más partes de la torta a su elección, sin que el otro jugador vea alterada su ganancia.

- La tabla de gustos **igual_vs_picky** resulta bastante similar a **opuesto_disjunto**. En este caso, el jugador 2 solo percibe un incremento en su utilidad si la torta contiene su gusto preferido, lo que deja al jugador 1 con muchas opciones de gustos para su mitad.

Con esto concluimos que el método con información completa en general es mejor que incompleta. Veamos ahora cómo se compara la ganancia total de cada jugador según las tablas de gustos, solo para información completa.

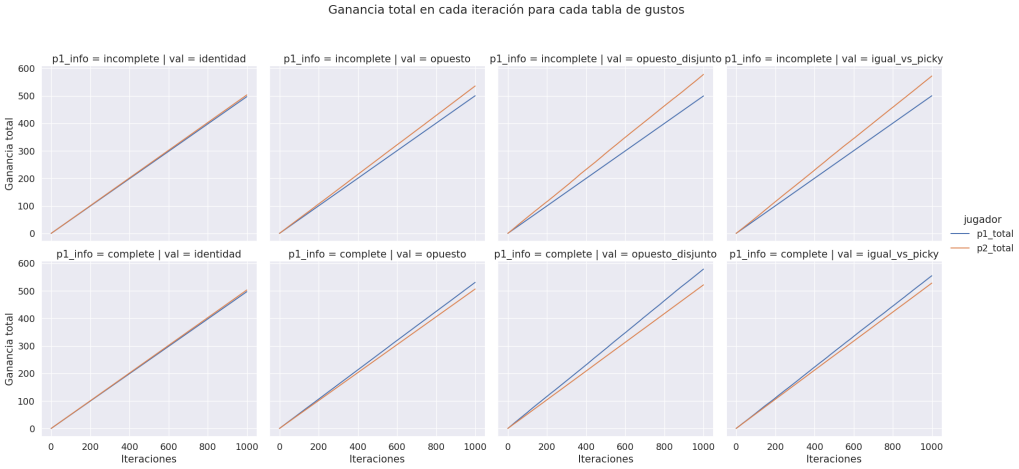

Figura 2: Ganancia acumulada en cada iteración de cada jugador según tabla de gustos

En la figura 2 podemos observar que el jugador 1 con información completa (segunda fila) logra sacar ventaja sobre el jugador 2. Esto no es así cuando el jugador 1 no conoce la función de utilidad del jugador 2 (para información incompleta, primera fila)

Conclusiones

Pudimos observar que, para el jugador encargado de cortar la torta, conocer la función de utilidad del otro jugador, resulta en una mayor ganancia. Sin embargo, qué tanta diferencia puede llegar a hacer depende bastante de las tabla de valoraciones.

Si bien la implementación que hicimos con fuerza bruta fue sencilla, queda como trabajo futuro pensar en implementaciones más eficientes, ya que para tortas muy grandes (por ejemplo de tamaño 1000) se tornaba muy lento evaluar todos los cortes.

También sería interesante probar aún más tablas de gustos, y contrastarlas con una estrategia aleatoria del jugador 1 como control.

Una conclusión obvia de esto es que al jugador encargado de elegir, no le conviene revelar su función de utilidad ya que el jugador encargado de cortar puede explotar sus gustos para beneficiarse. Un trabajo futuro posible podría ser agregar un segundo .ºráculo o espía"pero esta vez para el jugador que elige. De esta forma, este debería modificar su valoración de forma tal que, considerando que la van a intentar de explotar, este pueda sacarle ventaja a eso.