README
Задание №1

Выполнила
студентка 311 учебной группы факультета ВМК МГУ
Чернобай Анна Александровна

Москва
2021

Оглавление

Глава 1

Постановка задачи

Задание состоит в численном решении антагонистической матричной игры. В рамках данного задания необходимо:

- написать код, решающий матричную игру путем сведения ее к паре двойственных задач линейного программирования,
- проиллюстрировать работу данного кода путем визуализации спектров оптимальных стратегий,

Формально, задача заключается в следующем:

- (50 баллов) Необходимо написать функцию Nash_Equilibrium(a), которая принимает матрицу выигрыша и возвращает значение игры и оптимальные стратегии первого и второго игроков.
- (50 баллов) Проиллюстрировать работу вашего кода путем решения нескольких игр и визуализации спектров оптимальных стратегий игроков в Jupyter. В частности, нужно привести игры, в которых:
  1. спектр оптимальной стратегии состоит из одной точки (т.е. существует равновесие Нэша в чистых стратегиях),
  2. спектр оптимальной стратегии неполон (т.е. некоторые чистые стратегии не используются),
  3. спектр оптимальной стратегии полон.

Глава 2

Описание алгоритма

1. Проверяем есть ли в матрице седловые точки. Если седловые точки есть, то выписываем решение в чистых стратегиях. Значение игры - значение седловой точки.

2. Если седловых точек в матрице нет, то необходимо найти решение в смешанных стратегиях. Для этого сведем решение матричной игры к решениям двух задач линейного программирования следующим образом:

   Пусть антагонистическая игра задана платёжной матрицей А, имеющей размерность m×n

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

   Необходимо найти решение игры, т.е. определить оптимальные смешанные стратегии первого и второго игроков: $P = (p_1, p_2, \dots, p_m)$, $Q = (q_1, q_2, \dots, q_n)$ где P и Q - векторы, компоненты которых p_i и q_j характеризуют вероятности применения чистых стратегий i и j соответственно первым и вторым игроками и соответственно для них выполняются соотношения:

$$p_1 + p_2 + \ldots + p_m = 1$$

$$q_1 + q_2 + \ldots + q_n = 1$$

   Найдём сначала оптимальную стратегию первого игрока P. Эта стратегия должна обеспечить выигрыш первому игроку не меньше V, т.е. $\geq V$, при любом поведении второго игрока, и выигрыш, равный V , при его оптимальном поведении, т.е. при стратегии Q.
   Цена игры V пока неизвестна. Без ограничения общности, можно предположить её равной некоторому положительному числу $V > 0$. Действительно, для того, чтобы выполнялось условие $V > 0$, достаточно, чтобы все элементы матрицы A были неотрицательными. Этого всегда можно добиться с помощью аффинных преобразований: прибавляя ко всем элементам матрицы A одну и ту же достаточно большую положительную константу C (модуль минимального отрицального элемента в матрице, если таких нет, то берем $C = 0$); при этом цена игры увеличится на C, а решение не изменится. Итак, будем считать $V > 0$.
   Предположим, что первый игрок I применяет свою оптимальную стратегию P, а второй игрок

II свою чистую стратегию j-ю, тогда средний выигрыш (математическое ожидание) первого игрока будет равен:

$$a_{1j}p_1 + a_{2j}p_2 + \ldots + a_{mj}p_m = a_j$$

Оптимальная стратегия первого игрока I обладает тем свойством, что при любом поведении второго игрока II обеспечивает выигрыш первому игроку, не меньший, чем цена игры V; значит, любое из чисел a_j не может быть меньше V ($\geq V$). Следовательно, при оптимальной стратегии, должна выполняться следующая система неравенств:

$$\begin{cases} a_{11}p_1 + a_{21}p_2 + ... + a_{m1}p_m \geq V \\ a_{12}p_1 + a_{22}p_2 + ... + a_{m2}p_m \geq V \\ ... \\ a_{1n}p_1 + a_{2n}p_2 + ... + a_{mn}p_m \geq V \end{cases} \tag{2.1}$$

Разделим неравенства на положительную величину V (правые части системы) и введём обозначения:

$$x_1 = \frac{p_1}{V}; x_2 = \frac{p_2}{V}; ...; x_m = \frac{p_m}{V}$$
$$x_1 \geq 0; x_2 \geq 0; ...; x_m \geq 0$$

Тогда условия (2.1) запишутся в виде:

$$\begin{cases} a_{11}x_1 + a_{21}x_2 + ... + a_{m1}x_m \geq 1 \\ a_{12}x_1 + a_{22}x_2 + ... + a_{m2}x_m \geq 1 \\ ... \\ a_{1n}x_1 + a_{2n}x_2 + ... + a_{mn}x_m \geq 1 \end{cases} \tag{2.2}$$

В силу того, что $p_1 + p_2 + \ldots + p_m = 1$, следует

$$F = x_1 + x_2 + ... + x_m = \frac{1}{V} \tag{2.3}$$

Поскольку первый игрок I свой гарантированный выигрыш V старается сделать максимально возможным ($V \to$ max) , очевидно, при этом правая часть 2.3→min. Таким образом, задача решения антагонистической игры для первого игрока свелась к следующей математической задаче:

Определить неотрицательные значения переменных x_1, x_2, ... , x_m, чтобы они удовлетворяли системе функциональных линейных ограничений в виде неравенств 2.2, системе общих ограничений (все x_iнеотрицательные) и минимизировали целевую функцию F:

Это типичная задача линейного программирования и она может быть решена симплекс - методом. Таким образом, решая задачу линейного программирования, мы можем найти оптимальную стратегию $P = (p_1, p_2, \ldots, p_m)$ игрока I.

Найдём теперь оптимальную стратегию $Q = (q_1, q_2, \ldots, q_n)$ игрока II. Всё будет аналогично решению игры для игрока I, с той разницей, что игрок II стремиться не максимизировать, а минимизировать выигрыш I(по сути проигрыш II), а значит, не минимизировать, а максимизировать величину $\frac{1}{V}$, т.к. теперь $V \to$min. Вместо условий 2.2 должны выполняться условия:

$$\begin{cases} a_{11}y_1 + a_{12}y_2 + ... + a_{1n}y_n \leq 1 \\ a_{21}y_1 + a_{22}y_2 + ... + a_{2n}y_n \leq 1 \\ ... \\ a_{m1}y_1 + a_{m2}y_2 + ... + a_{mn}y_n \leq 1 \end{cases} \tag{2.4}$$

$$y_1 = \frac{q_1}{V}; y_2 = \frac{q_2}{V}; ...; y_n = \frac{q_n}{V}$$
$$y_1 \geq 0; y_2 \geq 0; ...; y_n \geq 0$$

Таким образом, задача решения антагонистической игры для второго игрока свелась к следующей математической задаче: определить неотрицательные значения переменных y_1, y_2, .., y_n, чтобы они удовлетворяли системе функциональных линейных ограничений в виде неравенств 2.4, системе общих ограничений (все y_iнеотрицательные) и максимизировать целевую функцию $F' = y_1 + y_2 + ... + y_n$:

Это типичная задача линейного программирования (прямая) и она может быть решена симплекс - методом

3. Необходимо привести задачи линейного программировния полученные в п.2 к такому виду, в котором они могли бы быть решены использованием linprog из библиотеки SciPy: ищется минимум целевой функции и в системе ограничений в неравествах не может быть использован знак $\geq$. Соответственно в задаче для первого игрока необходимо все неравенства умножить на -1, а для второго игрока умножить целевую функцию на -1.

4. Полученные с помощью linprog вектора оптимальных стратегий игроков I и II домножаем на $\frac{1}{player\_1.fun}$ (в player_1 возвращается результат работы linprog для игрока I). Затем получаем значение игры, вычитанием той величины C, которую мы прибавляли к матрице A на шаге 2, из $\frac{1}{player\_1.fun}$.

Глава 3

Описание функций программы

- `def saddle_point(A) -> tuple:`

  Функция принимает на вход матрицу A, в которой ищет максиминный и минимаксный элементы. Если их значения совпадают, т.е. это один и тот же элемент матрицы, то функция возвращает его индексы. В противном случае (значения не совпали) будет возвращено (None, None).

- `def nash_equilibrilium(A):`

  Функция принимает на вход матрицу A и возвращает вектора оптимальных стратегии 1-го и 2-го игроков p и q соответственно, значение игры game_value и тип стратегии strategy_type (чистая или смешанная). Сначала проверяем, есть ли седловые точки с помощью saddle_point(A). Если седловые точки есть, то обрабатываем результат работы saddle_point и возвращаем нужные значения. Если седловых точек нет, то решаются две задачи линейного программирования с помощью linprog из scipy.optimize, обрабатываем полученные результаты и возвращаем нужные значения

- `def print_matrix(A):`

  Функция принимает на вход матрицу A и печатает ее элементы с точностью до 3-х знаков после запятой.

- `def print_vector(v):`

  Функция принимает на вход вектор v и печатает его элементы с точностью до 3-х знаков после запятой.

- `def visualization(v):`

  Функция принимает на вход вектор v, по которому строится график "Визуализация спектров оптимальных стратегий".

- `def input_from_file(filename):`

  Функция принимает на вход имя файла filename и возвращает матрицу, считанную из файла filename.

- `def print_results(p, q, game_value, strategy_type):`

Функция принимает на вход вектора оптимальных стратегии 1-го и 2-го игроков p и q соответственно, значение игры game_value и тип стратегии strategy_type и печатает полученную информацию о матричной игре.

- `def main(file_name):`

  Функция принимает на вход имя файла filename. Сначала происходит вызов input_from_file и если считывание матрицы из файла прошло неуспешно, то работа функции завершается. В противном случае далее считанная матрица будет предана в функцию nash_equilibrilium и полученные в ходе работы данной функции вектора оптимальных стратегии 1-го и 2-го игроков p и q, значение игры game_value и тип стратегии strategy_type будут переданы на печать в print_results. Также будут визуализированы вектора p и q с помощью 2-х вызовов функции visualization.

Глава 4

Инструкция по запуску

Необходимые программы

- Python 3
- Jupyter

Необходимые библиотеки Python

- NumPy
- SciPy
- Matplotlib

Запуск

Далее считаем, что все выше перечисленные программы и библиотеки установлены.

- Для запуска программы необходимо скачать файл matrix_game.py. Далее матрицу, задающую антагонистическую матричную игру необходимо записать в txt-файл следующим образом: элементы отделяются друг от друга пробелами, каждая строчка записывается с новой строки (для записи матрицы в файл удобно воспользоваться savetxt из NumPy). Помещаем файл с матрицей и файл с программой в одну директорию, далее в этой директории открываем терминал и пишем python3 matrix_game.py имя файла с матрицей(например python3 matrix_game.py input.txt).
- Для запуска ноутбука необходимо запустить Jupyter и в нем открыть файл Task_1.ipynb

Литература

[1] Васин А.А., Краснощеков П.С., Морозов В.В. Исследование операций

[2] https://math.semestr.ru/games/linear-programming.php - описание сведения антагонистической матричной игры к двум задачам линейного программирования

[3] https://math.semestr.ru/games/gamesimplex.php - примеры матричных игр, часть из которых была использована в jupyter-notebook

[4] https://function-x.ru/games_matrix_games.html - примеры матричных игр, часть из которых была использована в jupyter-notebook

[5] https://www.hse.ru/mirror/pubs/share/183953534 - примеры матричных игр, часть из которых была использована в jupyter-notebook