Задание №1

Работа выполнена командой №4: задача состояла в численном решении антагонистической матричной игры. В рамках данного задания мы:

1. написали код, решающий матричную игру путем сведения ее к паре двойственных задач линейного программирования;
2. проиллюстрировали работу данного кода путем визуализации спектров оптимальных стратегий;
3. написали автоматические тесты для нашего решения.

Цель работы заключалась в том, чтобы познакомиться с языком программирования Python, библиотекой SciPy, интерактивной средой разработки Jupyter и системой тестирования Nose.

Содержание

1 Постановка задачи и дополнительная теория

Мы рассматриваем антагонистические матричные игры, наша задача — поиск значения игры и оптимальных стратегий двух игроков. Пусть функция $F(x, y)$ определена на декарстовом произведении $X \times Y$ где X, Y — множества произвольной природы.

1.1 Основные понятия

Определение. Пара $(x^0, y^0) \in X \times Y$ называется *седловой точкой* функции $F(x, y)$ на $X \times Y$, если

$$F(x, y^0) \leq F(x^0, y^0) \leq F(x^0, y) \; \forall x \in X, \forall y \in Y$$

или, эквивалентно,

$$\max_{x \in X} F(x, y^0) = F(x^0, y^0) = \min_{y \in Y} F(x^0, y).$$

Антагонистическая игра: в ней принимают участие два игрока 1 и 2. Игрок 1 выбирает стратегию x из множества X, игрок 2 выбирает стратегию y из множества стратегий Y. Нормальная форма игры подразумевает, что каждый игрок выбирает свою стратегию независимо, не зная выбора партнера. Задана *функция выигрыша* $F(x, y)$ первого игрока, определенная на $X \times Y$. Выигрыш $F(x, y)$ первого игрока является проигрышем для второго. Цель первого игрока состоит в увеличении своего выигрыша $F(x, y)$, а цель второго — в уменьшении $F(x, y)$. Таким образом, антагонистическая игра задается набором $\Gamma = \langle X, Y, F(x, y) \rangle$. Если значение $F(x, y) < 0$, то выигрыш первого игрока является фактически его проигрышем.

Смысл седловой точки: если игроки в качестве стратегий выбрали компоненты x^0, y^0 седловой точки, то каждому из них невыгодно отклоняться от выбранной стратегии. Поэтому седловая точка является формализацией концепции равновесия в игре.

Определение. Говорят, что антагонистическая игра Γ имеет решение, если функция $F(x, y)$ имеет на $X \times Y$ седловую точку. Пусть (x^0, y^0) - седловая точка функции $F(x, y)$. Тогда тройка (x^0, y^0, $v = F(x, y)$) называется решением игры, x^0, y^0 - *оптимальными* стратегиями игроков, а v — *значением* игры.

Определение. Антагонистическая игра Γ называется *матричной*, если множества стратегий игроков конечны: $X = \{1, \ldots, m\}$, $Y = \{1, \ldots, n\}$. При этом принято обозначать стратегию первого игрока через i, стратегию второго через j, а выигрыш первого $F(i, j)$ через a_{ij}. Матрица $\mathrm{A} = (a_{ij})_{m \times n}$ называется матрицей игры. Первый игрок выбирает в ней номер строки i, а второй — номер столбца j. В обозначениях матричной игры (i^0, j^0) — седловая точка матрицы A, если $a_{ij^0} \leq a_{i^0 j^0} \leq a_{i^0 j}$, $i = 1, \ldots, m, j = 1, \ldots, n$

Рассмотрим игру Γ с точки зрения первого игрока. Пусть он выбрал стратегию x. Ясно, что его выигрыш будет не меньше, чем $\inf_{y \in Y} F(x, y)$. Величину $\inf_{y \in Y} F(x, y)$ назовем *гарантированным результатом (выигрышем)* для первого игрока. Наилучший гарантированный результат для первого игрока $\underline{v} = \sup_{x \in X} \inf_{y \in Y} F(x, y)$ называется *нижним значением* игры.

Определение. Стратегия x^0 первого игрока называется *максиминной*, если $\inf_{y \in Y} F(x^0, y) = \underline{v}$.

Рассмотрим игру Γ с точки зрения второго игрока. Если он выбрал стратегию y, то для него естественно считать гарантированным результатом величину $\sup_{x \in X} F(x, y)$. Проигрыш второго игрока будет не больше, чем эта величина. Наилучший гарантированный результат для второго игрока $\overline{v} = \inf_{y \in Y} \sup_{x \in X} F(x, y)$ называется *верхним значением* игры.

Определение. Стратегия у второго игрока называется *минимаксной*, если $\sup_{x \in X} F(x, y^0) = \overline{v}$.

Если $\underline{v} = \overline{v}$, то говорят, что игра имеет седловую точку в чистых стратегиях, общее значение $\underline{v}$ и $\overline{v}$ называется при этом *ценой* игры и обозначается $V = \underline{v} = \overline{v}$. При этом стратегии игроков, соответствующие седловой точке, называются *оптимальными чистыми стратегиями*.

Смешанной стратегией первого игрока называется вектор $p = (p_1, \ldots, p_m)$, где все $p_i \geq 0$ $(i = 1, 2, ..., m)$, а $\sum_{i=1}^{m} p_i = 1$. При этом p_i — вероятность, с которой первый игрок выбирает свою i-ю стратегию. Аналогично определяется смешанная стратегия $q = (q_1, ..., q_n)$ второго игрока. Чистая стратегия также подпадает под определение смешанной — в этом случае все вероятности равны нулю, кроме одной, равной единице. Если игроки играют со своими смешанными стратегиями p и q соответственно, то математическое ожидание выигрыша первого игрока равно $M(p, q) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} p_i p_j$ (и совпадает с математическим ожиданием проигрыша второго игрока).

Стратегии p^* и q^* называются *оптимальными смешанными стратегиями* соответственно первого и второго игрока, если $M(p, q^*) \leq M(p^*, q^*) \leq M(p^*, q)$. Если у обоих игроков есть оптимальные смешанные стратегии, то пара (p^*, q^*) называется *решением* игры (или седловой точкой в смешанных стратегиях), а число $V = M(p^*, q^*)$ — *ценой* игры.

1.2 Суть метода

Алгоритм поиска решения матричной антагонистической игры, заданной матрицей выигрышей, имеющей размерность $m \times n$, сводится к алгоритму симплекс-метода решения пары взаимодействующих задач линейного программирования. Пусть антагонистическая игра задана матрицей выигрышей A, имеющей размерность $m \times n$. Необходимо найти решение игры, т.е. определить оптимальные смешанные стратегии первого и второго игроков: $p^* = (p_1^*, p_2^*, ..., p_m^*)$, $q^* = (q_1^*, q_2^*, ..., q_n^*)$, 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: $\sum\limits_{i=1}^{m} p_i^* = 1$, $\sum\limits_{j=1}^{n} q_j^* = 1$.

Найдём сначала оптимальную стратегию первого игрока p^*. Эта стратегия должна обеспечить выигрыш первому игроку не меньше V, при любом поведении второго игрока, и выигрыш, равный V, при его оптимальном поведении, т. е. при стратегии q^*. Цена игры V нам 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 M; при этом цена игры увеличится на M, а решение не изменится. Предположим, что первый игрок A применяет свою оптимальную стратегию p^*, а второй игрок B свою чистую стратегию j-ю, тогда средний выигрыш (математическое ожидание) первого игрока A будет равен: $a_j = a_{ij} p_1^* + ... + a_{mj} p_m^*$.

Оптимальная стратегия игрока A обладает тем свойством, что при любом поведении игрока B обеспечивает выигрыш первому игроку, не меньший, чем цена игры V; значит, любое из чисел a_j не может быть меньше V. Следовательно, при оптимальной стратегии, должна выполняться следующая система неравенств:

$$\begin{cases} a_{11}p_1^* + ... + a_{mj}p_m^* \geq V \\ a_{12}p_1^* + ... + a_{mj}p_m^* \geq V \\ ... \\ a_{1n}p_1^* + ... + a_{mn}p_m^* \geq V \end{cases}$$

Разделим неравенства на положительную величину V и введём обозначения $y_1, ..., y_m$ для новых переменных $\frac{p_1}{V}, ..., \frac{p_m}{V}, y_1 \geq 0, y_2 \geq 0, ..., y_m \geq 0$. Тогда условия запишутся в виде:

$$\begin{cases} a_{11}y_1 + ... + a_{mj}y_m \geq 1 \\ a_{12}y_1 + ... + a_{mj}y_m \geq 1 \\ ... \\ a_{1n}y_1 + ... + a_{mn}y_m \geq 1 \end{cases}$$

где $y_1, y_2, ..., y_m$ — неотрицательные переменные. В силу неравенств переменные $y_1, y_2, ..., y_m$ удовлетворяют условию, которое обозначим через F: $F = \sum\limits_{i=1}^{m} y_i = \frac{1}{V}$.

Поскольку первый игрок свой гарантированный выигрыш V старается сделать максимально возможным, очевидно, при этом правая часть — $\frac{1}{V} \to min$. Таким образом, задача решения антагонистической игры для первого игрока свелась к следующей математической задаче: определить неотрицательные значения переменных $y_1, y_2, ..., y_m$, чтобы они удовлетворяли системе функциональных линейных ограничений в виде неравенств, системе общих ограничений неотрицательности и минимизировали целевую функцию F: $\sum\limits_{i=1}^{m} y_i \to min$.

Это задача линейного программирования и она может быть решена симплекс-методом. Таким образом, решая задачу линейного программирования, можно найти оптимальную стратегию $p^* = (p_1^*, p_2^*, ..., p_m^*)$ игрока A. Чтобы найти оптимальную стратегию $q^* = (q_1^*, q_2^*, ..., q_n^*)$ игрока B, нужно провести аналогичные действия, с той разницей, что игрок B стремится не максимизировать, а минимизировать проигрыш, а значит, не минимизировать, а максимизировать величину $\frac{1}{V}$. Тогда должны выполняться условия:

$$\begin{cases} a_{11}x_1 + ... + a_{mj}x_m \leq 1 \\ a_{12}x_1 + ... + a_{mj}x_m \leq 1 \\ ... \\ a_{1n}x_1 + ... + a_{mn}x_m \leq 1 \end{cases}$$

где $x_1 = \frac{q_1}{V}, ..., x_m = \frac{q_m}{V}, x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0$.

Требуется так выбрать переменные $x_1, x_2, ..., x_n$, чтобы они удовлетворяли условиям и обращали в максимум линейную функцию цели F': $F' := \sum\limits_{i=1}^{n} x_i = \frac{1}{V} \to \max$.

Таким образом, задача решения антагонистической игры для второго игрока свелась к следующей математической задаче: определить неотрицательные значения переменных $x_1, x_2, ..., x_n$, чтобы они удовлетворяли системе функциональных линейных ограничений в виде неравенств, системе общих ограничений и максимизировали целевую функцию F'.

Это типичная задача линейного программирования и она может быть решена симплекс-методом. Таким образом, решая прямую задачу линейного программирования, мы можем найти оптимальную стратегию $Q^* = (q_1^*, q_2^*, ..., q_n^*)$ игрока B.

1.3 Алгоритм симплекс-метода

Алгоритм симплекс-метода заключается в том, что из множества вершин, принадлежащих границе множества решений системы неравенств, выбирается такая вершина, в которой значение целевой функции достигает максимума (минимума). По определенному правилу находится первоначальный опорный план (некоторая вершина области ограничений). Проверяется, является ли план оптимальным. Если да, то задача решена. Если нет, то переходим к другому улучшенному плану - к другой вершине. Значение целевой функции на этом плане (в этой вершине) заведомо лучше, чем в предыдущей. Алгоритм перехода осуществляется с помощью некоторого вычислительного шага, который удобно записывать в виде таблиц, называемых симплекстаблицами. Так как вершин конечное число, то за конечное число шагов мы приходим к оптимальному решению.

- I этап. Переход к канонической форме задачи линейного программирования путем введения неотрицательных дополнительных балансовых (базисных) переменных. Запись задачи в симплекс-таблицу. Между системой ограничений задачи и симплекс-таблицей взаимнооднозначное соответствие. Строчек в таблице столько, сколько равенств в системе ограничений, а столбцов - столько, сколько свободных переменных. Базисные переменные заполняют первый столбец, свободные - верхнюю строку таблицы. Нижняя строка называется индексной, в ней записываются коэффициенты при переменных в целевой функции. В правом нижнем углу первоначально записывается 0, если в функции нет свободного члена; если есть, то он записывается с противоположным знаком. На этом месте (в правом нижнем углу) будет значение целевой функции, которое при переходе от одной таблицы к другой должно увеличиваться по модулю.

- II этап. Проверка опорного плана на оптимальность. Для этого необходимо анализировать строку целевой функции F. Если найдется хотя бы один коэффициент индексной строки меньше нуля, то план не оптимальный, и его необходимо улучшить.

- III этап. Улучшение опорного плана. Из отрицательных коэффициентов индексной строки выбирается наибольший по абсолютной величине. Затем элементы столбца свободных членов симплексной таблицы делит на элементы того же знака ведущего столбца. Далее идет построение нового опорного плана. Переход к новому опорному плану осуществляется в результате пересчета симплексной таблицы методом Жордана–Гаусса.

- IV этап. Выписывание оптимального решения.

2 Описание программы

Функция linprog из библиотеки SciPy необходима для решения двойственной задачи линейного программирования с помощью симплекс-метода.

Макет функции linprog выглядит так:

`scipy.optimize.linprog(c, A_ub=None, b_ub=None, A_eq=None, b_eq=None, bounds=None, method='simplex', callback=None, options=None)`

Она решает следующую задачу линейного программирования с матрицей A: Минимизировать: $F = c^T * x$, Для системы уравнений $A_{ub} * x \leq b_{ub}$

Рассмотрим подробнее параметры функции:

- c — вектор коэффициентов для функции, которую необходимо минимизировать.

- A_{ub} — матрицы с коэффициентами для системы неравенств.
- b_{ub} — векторы правой части системы неравенств.

3 Требуемое ПО

3.1 Необходимые библиотеки

- SciPy — библиотека с открытым исходным кодом, предназначенная для выполнения научных и инженерных расчётов. Она необходима для функции linprog из пакета scipy.optimize, который содержит в себе множество алгоритмов оптимизации. Конкретно в нашей программе она не используется, но для решения задач ЛП всегда используют именно её.
- NumPy — библиотека с открытым исходным кодом с такими возможностями, как поддержка многомерных массивов (включая матрицы), поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.
- fractions – модуль, предоставляющий поддержку рациональных чисел — он содержит в себе функцию Fraction(), которая позволяет работать с обычными дробями.
- Nose — расширение над стандартными unittest, которое облегчает написание тестов.

3.2 Необходимые программы

- Python 3.6
- Anaconda3
- Jupyter

4 Инструкция по запуску

Здесь и далее будем считать, что пакет Anaconda установлен. В Anaconda при помощи Jupyter Notebook запускаем `python nash_IO.ipynb` и смотрим работу программы. Далее распаковываем zip-файл и в Anaconda с помощью Jupyter Notebook запускаем файл `python nash_IO.ipynb`, смотрим работу программы, которая собрана с помощью пакета pack, в котором находятся файлы с реализацией вспомогательных функций. Затем нужно открыть консоль, перейти в папку с проектом и написать команду `python nash_IO.py`.

Unit-тесты находятся в Notebook `python nash_IO.ipynb`, для их запуска в командной строке нужно будет ввести: `nosetests test_um_nash_eq.py`

5 Список участников и их вклад

- Максим Приходько — написание функции `nash_equilibrium`, визуализация спектров оптимальных стратегий и общая сборка
- Антон Тарасов — написание функции `nash_equilibrium`
- Кристина Светличная — написание readme и верстка данного файла в LaTeX

- Александра Коробельникова — оформление кода в виде пакета
- Екатерина Маслихина — написание unit-тестов