Постановка задачи

Задана матричная антагонистическая игра с матрицей выигрыша A. Требуется решить данную игру, привести примеры игр с различными спектрами оптимальных стратегий и проиллюстрировать решения графически.

Подход к решению

Определение игры и её решения

Антагонистическая игра $\Gamma = \langle X,\, Y,\, F = F(x,y)\rangle$ задаёт игру двух игроков с множествами стратегий X, Y и функцией выигрыша первого игрока $F(x,y)$. В такой игре функция выигрыша второго игрока $G(x,y) = -F(x,y)$.

Игра Γ называется матричной, если множества стратегий игроков конечны: $X = \{1,\ldots,m\}$, $Y = \{1,\ldots,n\}$. Матрица $A = (a_{ij})_{m\times n}$, в которой $a_{ij} = F(i,j)$, называется матрицей игры.

Чистой стратегией первого игрока называется стратегия i из множества X, которую он выбирает с вероятностью равной единице.

Смешанной стратегией первого игрока в игре Γ называется вероятностное распределение φ на множестве стратегий X. Тогда в матричной игре смешанной стратегией первого игрока будем называть вектор –

$$p = (p_1,\ldots,p_m) \in P = \{p \mid \sum_{i=1}^{m} p_i = 1, p_i \geq 0\},$$

смешанной стратегией второго игрока –

$$q = (q_1,\ldots,q_n) \in Q = \{q \mid \sum_{j=1}^{n} q_j = 1, q_j \geq 0\}.$$

Функция выигрыша первого игрока в смешанных стратегиях:

$$A(p,q) = \sum_{i=1}^{m}\sum_{j=1}^{n} p_i a_{ij} q_j.$$

Решение игры $(p_0,\, q_0,\, v = A(p_0,q_0))$ называется решением игры Γ в смешанных стратегиях, если

$$\max_{p\in P} A(p,q^0) = A(p^0,q^0) = \min_{q\in Q} A(p^0,q).$$

При этом p_0, q_0 называются оптимальными смешанными стратегиями игроков, а v – значением игры Γ.

По основной теореме матричных игр всякая матричная антагонистическая игра Г будет иметь решение в смешанных стратегиях. А значение игры может быть представлено в виде:

$$v = \max_{p \in P} \min_{1 \le j \le n} \sum_{i=1}^{m} p_i a_{ij} = \min_{q \in Q} \max_{1 \le i \le m} \sum_{j=1}^{n} a_{ij} q_j$$

Задача линейного программирования

Общей задачей линейного программирования (ЛП) называется задача, которая состоит в определении максимального (минимального) значения функции

$$z = \sum_{j=1}^{n} c_j x_j$$

при условиях

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i,\ i = 1, \ldots, k,\ x_j \ge 0.$$

Алгоритм решения задачи ЛП симплекс-методом:

1. Привести задачу к каноническому виду, заменяя все неравенства на уравнения с помощью введения новых переменных.

2. Записать систему уравнений, соответствующую данной задаче:

$$\begin{cases} z - c_1 x_1 - \ldots - c_n x_n = 0 \\ a_{i1} x_1 + \ldots + a_{in} x_n = b_i,\ i = 1, \ldots, m \end{cases}$$

3. Привести данную систему к диагональному виду по базисным переменным $z, x_1, \ldots, x_m$.

4. Составить таблицу коэффициентов (симплексная таблица):

	z	x_1	$\ldots$	x_m	x_{m+1}	$\ldots$	x_n
z	z_0	0	$\ldots$	0	c_{m+1}	$\ldots$	c_n
x_1	b_1	1	$\ldots$	0	$a_{1,m+1}$	$\ldots$	a_{1n}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
x_m	b_m	0	$\ldots$	1	$a_{m,m+1}$	$\ldots$	a_{mn}

5. Если все коэффициенты в первой строке при небазисных переменных $x_{m+1}, \ldots, x_n$ неотрицательны, то текущее базисной решение - оптимальное. Если это не так, то в число базисных переменных вводим переменную x_s, где номер s такой, что $c_s = \min\limits_{c_j < 0} c_j$.

6. Столбец с номером s – ведущий. Если в нём нет положительных коэффициентов, то оптимального решения не существует. В противном случае в качестве базисной переменной, которая исключается из числа базисных, выбирается та переменная x_r, для которой:

$$\frac{b_r}{a_{rs}} = \min_{a_{is}>0} \frac{b_i}{a_{is}}$$

7. a_{rs} называется ведущим элементом. Используя эквивалентные преобразования таблицы, пересчитываем таблицу так, чтобы ведущий элемент стал равным 1, а все остальные элементы ведущего столбца – равными 0.

8. Перейти к пункту 5 для исследования получившейся таблицы.

Сведение к задаче ЛП

Решим матричную игру путем сведения её к паре двойственных задач линейного программирования.

Будем предполагать, что значение игры положительно (этого всегда можно достичь увеличением элементов матрицы выигрыша на равное число). Введём новую переменную u и получим

$$v = \max_{(u,p)\in B} u, \text{ где } B = \{(u,p) \mid \sum_{i=1}^{m} p_i a_{ij} \geq u,\ j = 1,\ldots,n,\ \sum_{i=1}^{m} p_i = 1,\ p_i \geq 0\}.$$

Пусть $z_i = p_i/u$, тогда

$$v = \max_{(u,p)\in B} u = \frac{1}{\sum\limits_{i=1}^{m} z_i^0},$$

где вектор z_0 будем искать как оптимальное решение задачи линейного программирования

$$\sum_{i=1}^{m} z_i \to min$$

$$\sum_{i=1}^{m} a_{ij} z_i \geq 1,\ j = 1,\ldots,n,\ z_i \geq 0,\ i = 1,\ldots,m.$$

Применим описанный ранее алгоритм решения задачи ЛП и найдем вектор z^0. Далее по z^0 найдём значение игры и оптимальную смешанную стратегию первого игрока: $v = 1/\sum\limits_{i=1}^{m} z_i^0,\ p^0 = vz^0$.

Аналогично можно получить, что

$$v = \min_{q \in Q} \max_{1 \le i \le m} \sum_{j=1}^{n} a_{ij} q_j = \frac{1}{\sum_{j=1}^{n} w_j^0},$$

где w^0 – оптимальное решение задачи ЛП

$$\sum_{j=1}^{n} w_j \to max$$

$$\sum_{j=1}^{n} a_{ij} w_j \le 1,\ i = 1, \ldots, m,\ w_j \ge 0,\ j = 1, \ldots, n.$$

И тогда оптимальная смешанная стратегия второго игрока $q^0 = vw^0$.

Используемые средства python

Нами были использованы функции из библиотек Numpy, Scipy, Matplotlib.pyplot, Fractions, а так же стандартные функции языка Python3.

1. numpy (работа с матрицами и массивами)

   1.1 numpy.array(object).
   Создает массив, удобный для работы с остальными функциями данной библиотеки.

   1.2 numpy.around(a, decimals).
   Округляет число a до decimals знаков после запятой.

   1.3 matrix.Transpose().
   Транспонирует матрицу (объект numpy).

2. scipy (поиск седловой точки)

   2.1 scipy.optimize.linprog(c, A, b).
   Минимизирует $c^T x$ при условиях: $Ax \le b$. В поле x возвращает решение задачи в виде вектора коэффициентов.

3. matplotlib.pyplot (графическое представление решения)

   3.1 pyplot.plot([x], [y], [fmt]).
   Получает массив координат [x], координат [y], и формат [fmt]. Графически отображает линии между точками (x[1], y[1]) и (x[2], y[2]) и т.д. в указанном формате (цвет, наличие пунктира и т.п.). В случае указания формата 'o', отображает точки.

3.2 pyplot.axis(x1, x2, y1, y2).

Отображает масштаб на осях x и y, в пределах [x1, x2], [y1, y2] соответсвенно.

3.3 pyplot.show().

Выводит построенный график на экран.

4. fractions (работа с дробями)

4.1 Fraction(a).

Создает объект класса Fraction (рациональное число) из числа типа float.

4.2 Fraction.limit_denominator(b).

Возвращает объект Fraction, обыкновенную дробь близкую по значению к данной, но со знаменателем не больше, чем b.

5. Стандартные функции языка Python3 для работы с файлами и строками.

5.1 open(file_name, flag).

Открывает файл file_name с указанными флагами (на чтение/запись и т.п.)

5.2 String.replace(a, b).

Заменяет все вхождения подстроки a в строке на подстроку b.

5.3 String.strip(a).

Удаляет все символы a в начале и в конце строки.

5.4 String.split(a).

Возвращает следующий элемент отделенный в строке символом a.

5.5 print(a, end=b).

Выводит на экран объект a, в конце строки пишет символ b. По умолчанию, b = '\n'.

6. unittest (тестирование)

6.1 assertEqual(first, second, msg=None)

Проверяет на равенство first и second. Если эти значения не совпадают, то тест выдаёт ошибку AssertionError и выводится сообщение msg.

Реализация

1. Чтение матрицы из файла и приведение ее к удобному для работы виду.

   Открываем с помощью функции open() текстовый файл, содержащий матрицу в виде значений, разделенных символами '|'.

   Считываем матрицу построчно. С помощью функции replace удаляем все пробелы, а с помощью функции strip – символы '\n' и '|' из начала и конца строки. Извлекаем функцией split('|') числа из строки и преобразуем их к целому типу. Добавляем эти числа в буферный список, затем добавляем этот список в матрицу. С помощью numpy.array() приводим наш список к удобному для работы виду (объекту матрица из numpy).

2. Аналитическое решение игры. (nash_equilibrium(A_win))

   Если в матрице присутствуют отрицательные значения, то делаем все элементы матрицы положительными и запоминаем это действие.

   Сначала находим вектор оптимальной стратегии второго игрока. Создаем вектора коэффициентов для функции linprog(c, A, b). Вектор коэффициентов $c = [-1, \dots, -1]$, так как linprog решает задачу минимизации, а мы, исходя из теории, ищем максимум. Вектор $b = [1, \dots, 1]$. Применяем функцию linprog для решения нашей задачи. Вычисляем значение игры, делаем обратную замену переменных, получаем вектор оптимальных стратегий второго игрока для исходной игры.

   Для первого игрока транспонируем матрицу, умножая все её значения на -1 и создаем новые вектора коэффициентов $c = [1, \dots, 1]$, $b = [-1, \dots, -1]$. Матрица A и вектор b умножаются на -1, так как ограничения для первого игрока имеют вид $Ax \geq b$, а linprog решает задачу с ограничениями $Ax \leq b$. Применяем linprog и получаем вектор оптимальной стратегии первого игрока.

   Проверяем, пришлось ли нам делать элементы положительными. Если да, то пересчитываем значение игры.

3. Графическое представление векторов оптимальных стратегий (visualisation(vect))

   Создаем вектор ординат всех точек. С помощью pyplot.plot отображаем линии от оси OX ко всем точкам. С помощью pyplot.plot отображаем все точки. Используя pyplot.axis отображаем масштаб на осях и выводим наш график на экран.

4. Вывод

   С помощью print выводим наши вектора и значение игры на экран.

5. Тестирование программы

   Для проверки правильности работы программы используется встроенный в Python модуль unittest, который поддерживает автоматизацию тестов, использование общего кода для настройки и завершения тестов, объединение тестов в группы. Для автоматизации тестов используется концепция тестовый случай (test case) - минимальный блок тестирования. Он проверяет ответы для разных наборов данных. Модуль unittest предоставляет базовый класс TestCase, который можно использовать для создания новых тестовых случаев. Тестовый случай создаётся путём наследования от unittest.TestCase. В нём определяется единственная функция для проверки тестов. Для сравнения и проверки ожидаемых результатов используется функция assertEqual. Функция all() используется, чтобы проверить, что все значения последовательности равны True. Инструкция unittest.main() предоставляет интерфейс командной строки для тестирования программы. Будучи запущенным из командной строки, этот скрипт выводит отчёт о результате запуска тестов.

Вклад участников группы

- Арбузов П.А.

  Реализация графической интерпретации решения, написание файла readme (реализация), помощь в редактировании кода, написании комментариев и устранении ошибок.

- Кюнченкова Д.Д.

  Реализация основной функции nash_equilibrium(), реализация графической интерпретации решения, написание файла readme (реализация), написание тестов, помощь в редактировании кода, написании комментариев и устранении ошибок.

- Семенов А.В.

  Написание файла readme (теория), реализация чтения матрицы из файла и приведения ее к удобному для работы виду, реализация вывода решения, помощь в редактировании кода, написании комментариев и устранении ошибок.