Численное решение антагонистической матричной игры.

I) Постановка задачи:

Задание состоит в численном решении антагонистической матричной игры, визуализации спектров оптимальных стратегий в Jupiter, написания unit-тестов и оформления решения в виде пакета.

Матрица $A = (a_{ij})_{m \times n}$ называется матрицей игры, в которой первый игрок выбирает номер строки, а второй - номер столбца.

Задача:

По матрице выигрыша построить векторы стратегий игроков и найти цену игры (рациональное число). Будем считать, что стратегия первого игрока заключается в минимизации выигрыша, а стратегия второго в максимизации.

Алгоритм:

1. Решение в чистых стратегиях:

- 1.1 Если матрица имеет седловую точку, мы можем выписать решение в чистых стратегиях.

2. Решение в смешанных стратегиях:

- 2.1 Если в матрице присутствуют отрицательные элементы. Для упрощения расчетов добавим к элементам матрицы модуль самого маленького числа. Такая замена не изменит решения игры, изменится только ее цена (по теореме фон Неймана).

- 2.2 Сведем задачу к задаче линейного программирования. Решим прямую задачу линейного программирования симплексным методом, с использованием симплекной таблицы. Систему неравенств приведем к системе уравнений путем введения дополнительных переменных. Полученная система будет записана в канонической форме.

- 2.3 Построим первичную симплекс-таблицу. За первичный базис возьмем дополнительные переменные. Индексная строка (последняя строка симплекс-таблицы) принимает значение -1 в столбцах основных переменных и 0 в столбцах дополнительных переменных.

- 2.4 Будем переходить от базиса к базису пока все значения индексной строки не станут положительными. Если в индексной строке есть отрицательные значения, то опорный план не оптимален и нам нужно построить новую таблицу с новыми базисными переменными y_s.

  Алгоритм поиска базисных переменных:

  – 2.4.1 Выберем ведущий столбец (column), в котором значение индексной строки (row) минимально (при равенстве значений выберем последний столбец).

  – 2.4.2 Столбец значений поэлементно поделим на значения ведущего столбца. Наименьшее значение определяет ведущую строку.

  – 2.4.3 На пересечении ведущего столбца и ведущей строки получим ведущий элемент (elem).

  – 2.4.5 Базисная переменная, соответствующая ведущему столбцу заменит базисную перерменную, соответствующйю ведущей строке.

- 2.5 Преобразуем таблицу.

  – 2.5.1 Разделим каждый элемент ведущей строки на elem:

$$a_{rowj} = \frac{a_{rowj}}{elem}$$

  – 2.5.2 Выберем строку i и столбец j. Вычтем из элемента a_{ij} произведение элемента i строки ведущего столбца на элемент j столбца ведущей строки.

$$(a_{ij}) = a_{ij} - a_{icolumn} * a_{rowj}$$

- 2.6 Если в индексной строке по прежднем есть отрицательные значения перейти к пункту 2.4.

- 2.7 Построение вероятностных векторов.

  – 2.7.1 Для первого игрока: вектор, состоящий из значений столбца значений, соответствующих основным переменным в порядке их индексирования.Если переменная отсутствует - вероятность, соответствующая этой перерменной, принимает значение ноль.

  – 2.7.2 Для второго: вектор, состоящий из значений индексной строки, соответствующих дополнительным переменным в порядке их индексирования. Если переменная отсутствует - вероятность, соответствующая этой перерменной, принимает значение ноль.

– 2.7.3 Дробь, обратная сумме значений вектора - цена игры. Умножим каждый вектор на цену игры и получим вероятностные векторы.

II) Требования к системе:

1)Python версии не ниже 3.6

2)Необходимые пакеты: - matplotlib - numpy

3)Запуск nash_equilibrium: На вход подается матрица игры. На выходе: p-стратегия первого игрока q-стратегия второго игрока price-цена игры flag-индикатор седловых точек print_result: На вход подается параметры: s-стратегия первого игрока r-стратегия второго игрока p-цена игры f-индикатор седловых точек На выходе: Выводит значения этих пременных и строит визуализацию спектров.

from nash_equilibrium.simplex_method import nash_equilibrium, print_result
tab = np.array([[4,0,6,2,2,1], [3,8,4,10,4,4], [1,2,6,5,0,0], [6,6,4,4,10,3], [10,4,6,4,0,9], [10,7,0,7,9,8]])
p, q, price, flag = nash_equilibrium(tab)
print_result(p, q, price, flag)

III) Список участников с вкладом каждого:

Александра Липатова: Создание ветки, unit-тесты, оформление в виде пакета, ReadMe-II.
Вера Разумова: Визуализация спектров, Readme I, Readme III, поиск седловых точек.
Михаил Смирнов: Решение задачи симплекс методом, отладка и тестирование, Readme I.