README
Задание 1

Составила: Бурамбекова А.Н.(312 гр.)

1 Постановка задачи

Задание:

-выполнить численное решение антагонистической матричной игры.

-написать код, решающий матричную игру путем сведения ее к паре двойственных задач линейного программирования;

-проиллюстрировать работу данного кода путем визуализации спектров оптимальных стратегий;

-написать автоматические тесты для решения.

Цель: Познакомиться с языком программирования Python, библиотекой SciPy, интерактивной средой разработки Jupyter и с системой тестирования Nose.

2 Описание задачи и решения

Матричная игра — конечная игра двух игроков с нулевой суммой. Предположим, что первый игрок имеет m стратегий A_i , i = [1,m] , а второй игрок n стратегий B_j , j = [1,n] . Тогда игра может быть названа игрой m × n или m × n игрой. Обозначим через a_{ij} значения выигрышей игрока A (соответственно — значения проигрышей игрока B), если первый игрок выбрал стратегию A_i, а второй игрок стратегию B_j. В этом случае говорят, что имеет место ситуация A_i, B_j. Значения выигрышей a_{ij} (эффективностей) можем представить в виде платежной таблицы, называемой матрицей игры или платежной матрицей:

$$A = \left\{ a_{ij} \right\} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Рассмотрим игру двух лиц, интересы которых противоположны. Такие игры называют антагонистическими играми двух лиц. В этом случае выигрыш одного игрока равен проигрышу второго, и можно описать только одного из игроков. Предполагается, что каждый игрок может выбрать только одно из конечного множества своих действий. Выбор действия называют выбором стратегии игрока. Если каждый из игроков выбрал свою стратегию, то эту пару стратегий называют ситуацией игры. Следует заметить, каждый игрок знает, какую стратегию выбрал его противник, т.е. имеет полную информацию о результате выбора противника.

Чистой стратегией игрока I является выбор одной из n строк матрицы выигрышей A, а чистой стратегией игрока II является выбор одного из столбцов этой же матрицы.

2.1 Решение в чистых стратегиях

1. Проверка на наличие седловой точки у платежной матрицы. Если да, то выписываем решение игры в чистых стратегиях. Считаем, что игрок I выбирает свою стратегию так, чтобы получить максимальный свой выигрыш, а игрок II выбирает свою стратегию так, чтобы минимизировать выигрыш игрока I.

В каждом столбце платежной матрицы определяем максимальный элемент и среди них находим минимальный - minmax(минимакс). Пусть этому элементу соответствует чистая стратегия B_j игрока B.

В каждой строке платежной матрицы определяем минимальный элемент и среди них находим максимальный - maxmin(максимин). Пусть этому элементу соответствует чистая стратегия A_i игрока A .

Если эти два числа равны, то седловая точка для данной матрицы существует. Стратегии A_i и B_j являются оптимальными стратегиями. Решение найдено. Данная ситуация называется равновесная ситуация A_i, B_j.

```
for i in range(len(A)): # ищем седловую точку как нижнее и верхнее значение игры
        game_val = min(A[i])
        index_map = filter(lambda x: A[i][x] == game_val, range(len(A[i])))
        for j in list(index_map):
```

```
                for i in range(len(A)):
                        if A[i][j] > game_val:
                                break
                        else:
                                p[i] = 1
                                q[j] = 1
                                pure = True
                                return p, q, game_val, pure
    if not pure:
            p, q, game_val = simplex(A) #если седловой точки нет, ищем решение в смешанных страте
            return p, q, game_val, pure
```

2.2 Решение в смешанных стратегиях

1.Если в матрице есть отрицательные элементы, избавимся от них добавив к каждому ее элементу модуль наименьшего элемента.

2.Сведем задачу к задаче линейного программирования.Решим прямую задачу линейного программирования симплексным методом с использованием симплексной таблицы. Введением дополнительных переменных приведем систему неравенств к системе равенств.

3.Построим первичную симплекс-таблицу. В качестве первичного базиса возьмем дополнительные переменные. Индексная строка принимает значение -1 в столбцах основных переменных и 0 в столбцах дополнительных переменных.

4.Будем переходить от базиса к базису до тех пор, пока все значения индексной строки не станут положительными.Опорный план не станет оптимальным пока в индексной строке есть отрицательные значения. Надо построить новую таблицу с новыми базисными переменными y_s

Алгоритм поиска базовых переменных:

•Выберем ведущий столбец (c), в котором значение индексной строки (r) минимально.

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

•На пересечении ведущего столбца и ведущей строки находится ведущий элемент a^*.

5.Разделим каждый элемент ведущей строки на a^* Зафиксируем i, j. Вычтем из a_{ij} произведение $a_{ic}a_{rj}$. Если в индексной строке остались отрицательные элементы, повторяем пункт 4.

6. Построение вероятностных векторов

- Для первого игрока

вектор, состоящий из значений столбца значений, соответствующих основным переменным в порядке их индексирования. Если переменная отсутствует , то вероятность, которая ей соответствует, равна 0.

- Для второго игрока вектор, состоящий из значений индексной строки, соответствующих дополнительным переменным в порядке их индексирования. Если переменная отсутствует , то вероятность, которая ей соответствует, равна 0.

-Число, которое является обратным к сумме значений вектора - значение игры. Умножим каждый вектор на значение игры и получим вероятностные векторы.

3 Запуск тестов

pytest tests.py