**Что такое антагонистическая матричная игра?**

Рассмотрим конфликт двух участников с противоположными интересами, математической моделью которого является игра с нулевой суммой. Участники игры – лица, принимающие решения, – называются игроками. Стратегия игрока – это осознанный выбор одного из множества возможных вариантов его действий.

Будем рассматривать конечные игры, в которых множества стратегий игроков конечны; стратегии первого игрока пронумеруем числами от 1 до m, а стратегии второго игрока — числами от 1 до n.

Если первый игрок выбрал свою i-ю стратегию, а второй игрок – свою j-ю стратегию, то результатом такого совместного выбора будет платеж aij второго игрока первому (это не обязательно денежная сумма, а любая оценка полезности результата выбора игроками своих стратегий i и j). Таким образом, конечная игра с нулевой суммой однозначно определяется матрицей (aij), i = 1,…, m, j = 1, …, n, которая называется платежной матрицей (или матрицей выигрышей). Строки этой матрицы соответствуют стратегиям первого игрока, а столбцы – стратегиям второго игрока. Конечные игры с нулевой суммой называются матричными, так как целиком определяются своими платежными матрицами.

Игра происходит партиями. Партия игры состоит в том, что игроки одновременно называют свой выбор: первый игрок называет некоторый номер строки матрицы A (по своему выбору или случайно), а второй – некоторый номер столбца этой матрицы (также по своему выбору или случайно). После этого происходит «расплата».

Пусть, например, первый игрок назвал номер i, а второй – j. Тогда второй игрок платит первому сумму aij. На этом партия игры заканчивается. Если aij > 0, то это означает, что при выборе первым игроком i-й стратегии, а вторым j-й выигрывает первый игрок; если же aij < 0, то это значит, что при данном выборе стратегий в выигрыше оказывается второй игрок. Цель каждого игрока – выиграть как можно большую сумму в результате большого числа партий.

Смысл названий «конфликт с противоположными интересами» и «игра с нулевой суммой» состоит в том, что выигрыш каждого из игроков противоположен выигрышу противника, или, иначе, что сумма выигрышей игроков равна нулю. Стратегия называется чистой, если выбор игрока неизменен от партии к партии. У первого игрока, очевидно, есть m чистых стратегий, а у второго – n. При анализе игр противник считается сильным, т.е. разумным.

Рассмотрим описанную конфликтную ситуацию с точки зрения первого игрока. Если первый игрок выбирает свою i-ю стратегию (i-ю строку матрицы A), то второй игрок, будучи разумным, выберет такую стратегию j, которая обеспечит ему наибольший выигрыш (а первому игроку, соответственно, наименьший), т. е. он выберет такой столбец j матрицы A, в котором платеж aij (второго игрока первому) минимален. Переберем все наши стратегии i = 1, 2, …, m и выберем ту из них, при которой второй игрок, действуя максимально разумно, заплатит первому игроку наибольшую сумму. Величина

α = max по i = 1,…,m min по j=1,…,n aij

называется нижней ценой игры, а соответствующая ей стратегия первого игрока – максиминной. Аналогичные рассуждения (но уже с точки зрения второго игрока) определяют верхнюю цену игры

β = min по i = 1,…,m max по j=1,…,n aij

и соответствующую ей минимаксную стратегию второго игрока. *Нижняя цена игры α представляет собой максимальный гарантированный выигрыш первого игрока* (т. е. применяя свою максиминную стратегию, первый игрок обеспечивает себе выигрыш, не меньший α), а *верхняя цена – величину, противоположную минимальному гарантированному проигрышу второго игрока* (т.е. применяя cвою минимаксную стратегию, второй игрок гарантирует, что он не проиграет больше чем β, или, по-другому, выиграет не меньше чем (–β)). Если α = β, то говорят, что игра имеет седловую точку в чистых стратегиях, общее значение α и β называется при этом ценой игры и обозначается V = α = β. При этом стратегии игроков, соответствующие седловой точке, называются оптимальными чистыми стратегиями, так как эти стратегии являются наиболее выгодными сразу для обоих игроков, обеспечивая первому игроку гарантированный выигрыш не менее V, а второму игроку – гарантированный проигрыш не более (–V), и отклоняться игрокам от этих стратегий невыгодно.

Смешанной стратегией первого игрока называется вектор p = (p1, …, pm), где все pi >= 0 (i=1, 2 ,…, m), а ∑pi = 1. При этом pi – вероятность, с которой первый игрок выбирает свою i-ю стратегию. Аналогично определяется смешанная стратегия q = (q1, …, qn) 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) = ∑∑aijpiqi

(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\*) <= M(p\*, q\*) <= 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.

**Суть метода:**

Алгоритм поиска решения матричной антагонистической игры, заданной матрицей выигрышей, имеющей размерность m×n, сводится к алгоритму симплекс-метода решения пары взаимодвойственных задач линейного программирования. Пусть антагонистическая игра задана матрицей выигрышей A, имеющей размерность m×n. Необходимо найти решение игры, т.е. определить оптимальные смешанные стратегии первого и второго игроков: p\*= (p1\*, p2\*, …, pm\*), q\*= (q1\*, q2\*,…, qn\*), где P\* и Q\* - векторы, компоненты которых pi\* и pj\* характеризуют вероятности применения чистых стратегий i и j соответственно первым и вторым игроками и соответственно для них выполняются соотношения:

p1\* + p2\* + … + pm\* = 1,

q1\* + q2\* + … + qn\* = 1

Найдём сначала оптимальную стратегию первого игрока p\*. Эта стратегия должна обеспечить выигрыш первому игроку не меньше V, т.е. ≥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 М; при этом цена игры увеличится на М, 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:

aj = aijp1\* + … + amjpm\*

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

a11p1\* + … + amjpm\* >= V

a12p1\* + … + amjpm\* >= V

…

a1np1\* + … + amnpm\*>= V

(1)

Разделим неравенства (1) на положительную величину V (правые части системы) и введём обозначения y1, …, ym для новых переменных p1/V, …, pm/V, y1 ≥ 0, y2 ≥ 0, .., ym ≥ 0. (2) Тогда условия запишутся в виде:

a11y1 + … + amjym >= 1

a12y1 + … + amjym >= 1

…

a1ny1 + … + amnym >= 1

(3)

где y1, y2, ..., ym - неотрицательные переменные. В силу (2) переменные y1, y2 , ..., ym удовлетворяют условию, которое обозначим через F:

F = y1 + y2 + … + ym = 1/V.

Поскольку первый игрок свой гарантированный выигрыш (V) старается сделать максимально возможным (V → max) , очевидно, при этом правая часть – 1/V → min - принимает минимальное значение. Таким образом, задача решения антагонистической игры для первого игрока свелась к следующей математической задаче: определить неотрицательные значения переменных y1, y2, ..., ym, чтобы они удовлетворяли системе функциональных линейных ограничений в виде неравенств (3), системе общих ограничений (2) и минимизировали целевую функцию F:

F = y1 + y2 + ... + ym → min

Это задача линейного программирования (двойственная) и она может быть решена симплекс-методом. Таким образом, решая задачу линейного программирования, можно найти оптимальную стратегию p\*= (p1\*, p2\*, …, pm\*) игрока A. Чтобы найти оптимальную стратегию q\*= (q1\*, q2\*,…, qn\*) игрока B, нужно провести аналогичные действия, с той разницей, что игрок B стремится не максимизировать, а минимизировать выигрыш (по сути - проигрыш), а значит, не минимизировать, а максимизировать величину 1/V, т.к. V → min. Вместо условий (3) должны выполняться условия:

a11x1 + … + amjxm <= 1

a12x1 + … + amjxm <=1

…

a1nx1 + … + amnxm <= 1

(4)

где x1 = q1/V, …, xm = qm/V,

x1 ≥ 0, x2 ≥ 0, .., xn ≥ 0. (5)

Требуется так выбрать переменные x1, x2, .., xn, чтобы они удовлетворяли условиям (4), (5) и обращали в максимум линейную функцию цели F':

F’ = x1 + x2 + … + xm = 1/V → max

Таким образом, задача решения антагонистической игры для второго игрока свелась к следующей математической задаче: определить неотрицательные значения переменных x1, x2, .., xn, чтобы они удовлетворяли системе функциональных линейных ограничений в виде неравенств (4), системе общих ограничений (5) и максимизировать целевую функцию F':

F' = x1 + x2 + .. + xn → max

Это типичная задача линейного программирования (прямая) и она может быть решена симплекс-методом. Таким образом, решая прямую задачу линейного программирования, мы можем найти оптимальную стратегию Q\*= (q1\*, q2\*,…, qn\*) игрока B.

**В чём заключается алгоритм симплекс-метода?**

Алгоритм симплекс-метода заключается в том, что из множества вершин, принадлежащих границе множества решений системы неравенств, выбрается 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 симплекс-таблицами. Так как вершин конечное число, то за конечное число шагов мы приходим к оптимальному решению.

**Этапы:**

1. 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 взаимно-однозначное 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.
2. 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.
3. 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.
4. IV этап. Выписывание оптимального решения.

**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 c матрицей A:

Минимизировать:

F = c^T \* x,

Для системы уравнений

A\_ub \* x <= b\_ub

A\_eq \* x == b\_eq

Рассмотрим подробнее параметры функции:

* с - это вектор коэффициентов для функции, которую необходимо минимизировать. В нашем случае это единичный вектор.
* A\_ub, A\_eq
* Матрицы с коэффициентами для системы неравенств, равенств соответственно. Так как системы (3), (4) состоят лишь из неравенств, то на вход функции мы не подаем матрицу A\_eq.
* b\_ub, b\_eq Векторы правой части системы неравенств, равенств соответственно. Для нашей задачи b\_ub – единичный вектор, а так как системы (3), (4) состоят лишь из неравенств, то на вход функции мы не подаем вектор b\_eq.
* bounds : sequence, optional Определяет границы для переменных xi. В нашей программе мы определяем неотрицательные границы (0, None) для каждого xi в связи с условиями задачи.
* method : str, optional Тип метода, используемый для решения задачи. В данном случае мы используем симплексный метод (в настоящий момент только он и поддерживается).
* callback : callable, optional Функция обратного вызова. Она может быть вызвана для каждой итерации симплексного алгоритма, например, для получения вектора решений для текущей итерации. В данной программе эта функция не используется.
* options : dict, optional Опции. Можно задать максимальное количество итераций, разрешение печати сообщения о результате работы.

**Ход программы:**

1. Печатаем матрицу выигрышей.
2. Сначала решаем первую задачу для первого игрока, описанную в сути метода. Создаем единичные векторы c и b длины первой строки и первого столбца матрицы соответственно. Это векторы np\_array, для создания которых используется библиотека numpy. Создаем вектор limits с границами (0, None) для каждой переменной xi, i = 1, …, m.
3. Для нахождения стратегии P первого игрока мы подаем на вход функции linprog транспонированную матрицу со знаком минус. Для транспонирования используется функция numpy.transpose(ar).
4. Так как система (3) представляет собой систему со знаком “>=”, вектор b подаем на вход со знаком минус.
5. Получаем результат работы функции linprog в переменную res типа scipy.optimize.OptimizeResult, печатаем найденную цену игры V (res.fun), а так же оптимальную стратегию первого игрока – вектор p, который получаем из x (res.x) путем деления на V.
6. Для печати используем модуль Fractions для вывода рациональных дробей, а так же функцию limit\_denominator(1000), чтобы выводить дробь со знаменателем 1000, максимально приближенную к данному числу float.
7. Для поиска оптимальной стратегии q второго игрока, подаем на вход функции linprog матрицу a, вектор v2, вектор границ limits, и вектор c со знаком минус, т. к. необходимо максимизировать функцию F’.
8. Чтобы получить вектор q, необходимо поделить элементы вектора x на (-res.fun), так как из-за максимизации функции мы получили цену игры со знаком минус.
9. Далее выводим значения вектора q с помощью модуля Fractions.

По ходу программы в функции nash\_equilibrium(ar) вызывается функция build\_graf(x, y).

Внутри функции build\_graf вызывается команда plt.figure()(добавление на рисунок прямоугольной (по умолчанию) области рисования), команда plt.scatter()(она ставит маркер значений вектора оптимальных стратегий) вызывается в цикле, команда plt.grid() строит линии вспомогательной сетки, plt.show() показывает рисунок.

**Необходимое ПО**

Windows 7

**Необходимые программы (библиотеки):**

*Библиотеки для nash\_equilibrium(a):*

• 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. Цель функции linprog - минимизация линейной объективной функции с линейными ограничениями равенства и неравенства.

• NumPy — библиотека с открытым исходным 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. Нам он необходим для того, чтобы выводить дроби.

*Для второй части задания:*

• Matplotlib Для визуализации решений

• UnitTest Для тестирования функций

**Программы:**

Python 3.6

Anaconda3

Jupyter

**Инструкция к запуску**

Здесь и далее будем считать, что пакет Anaconda установлен в Windows, в папку C:\Anaconda3, в Linux, вы его можно найти в каталоге, который выбрали при установке.

Перейдите в папку Scripts и введите в командной строке: ipython notebook

Если вы находитесь в Windows и открыли папку C:\Anaconda3\Scripts через проводник, то для запуска интерпретатора командной строки для этой папки в поле адреса введите cmd.

В результате запустится веб-сервер и среда разработки в браузере.

Выбираем файл task1.ipynd и запускаем его.