Домашнее задание №1

Иванов Вячеслав, группа 699

6 октября 2018 г.

Оглавление

1 Посели меня, если сможешь

Пусть n студентов надо расселить в m комнатах, $m < n$, в каждой комнате могут одновременно жить максимум 3 студента. Предпочтение студента i жить со студентом j задано и равно p_{ij}. Чем выше p_{ij}, тем более охотно студент i будет жить со студентом j. Также известно, что удовлетворённость студента i от проживания в комнате k равна b_{ik} и чем она выше, тем более охотно студент будет там проживать. Очевидно, что каждый студент может быть поселён только в одну комнату. Необходимо определить кого в какую комнату необходимо поселить? Рассмотрите случай, когда известны не все значения p_{ij}. Как изменится постановка задачи в этом случае?

1.1 Все p_{ij} известны

Условие можно переписать в виде задачи целочисленного квадратичного программирования. Здесь и далее будем считать, что $p_{ij} \in [0, 1]$ (т.к. в таком виде с ними удобнее работать и к нему всегда можно привести, поделив на $(\max\limits_{i,j} p_{ij} - \min\limits_{i,j} p_{ij})$).

$$\max_{W \in \{0,1\}^{n \times m}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} p_{ij} \sum_{k=1}^{m} w_{ik} w_{jk} + \sum_{i=1}^{n} \sum_{k=1}^{m} w_{ik} b_{ik}$$

$$s.t. \begin{cases} \sum_{k=1}^{m} w_{ik} = 1, & \forall 1 \leqslant i \leqslant n \\ \sum_{i=1}^{n} w_{ik} \leqslant 3, & \forall 1 \leqslant k \leqslant m \end{cases}$$

Где W — матрица поселения: $w_{ik} \in \{0, 1\}$ говорит, поселили ли студента i в комнату k.
Область поиска — 2^{nm} булевых матриц.
Первая сумма в целевой функции отражает предпочтения студентов по поводу соседей, вторая — предпочтения по поводу комнат.
Первое ограничение — "каждого студента нужно поселить в какую-то комнату, и только в одну".
Второе ограничение — "в одной комнате не может проживать более 3-х студентов".
Целевую функцию нужно взять со знаком минус и минимизировать, чтобы привести задачу к стандартному виду.

1.2 Некоторые p_{ij} неизвестны

Кажется, что достаточно положить неизвестные значения равными нулю, и тогда вначале будут селить тех, кому точно комфортно вместе, а только потом уже тех, про кого ничего не ясно. Т.е. модель останется прежней. Но доказать я это пока не могу.

2 Перевозчик

Пусть на i-ом складе сети из n располагается a_i единиц товара, который необходимо развезти по m магазинам сети. Каждому j-ому магазину необходимо b_j единиц товара. Перевозка единицы товара из i-го склада в j-ый магазин стоит c_{ij} и занимает t_{ij} единиц времени. Сколько единиц товара с каждого склада нужно вывезти, чтобы обеспечить товаром все магазины сети?

Сразу отметим, что в условии не указано:

- Что именно следует оптимизировать: время или стоимость доставки или их взвешенную сумму с некоторыми коэффициентами?
- Нужно ли доставлять товар ежедневно или заказ одиночный?

- Сколько машин на каждом из складов и какова вместимость каждой из них?
- Единицы товара непрерывные (килограммы) или дискретные (упаковки)?

Каждый из перечисленных пунктов существенно меняет задачу. В самой простой формулировке заказ одиночный, время на доставку, количество и вместимость машин не ограничены.
Обозначим через $x_{ij} \in \mathbb{R}^+$ количество товара, которое будет доставлено со склада i в магазин j. Выпишем задачу оптимизации, используя матрицу X:

$$\min_{X \in (\mathbb{R}^+)^{n \times m}} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}(\alpha c_{ij} + \beta t_{ij})$$
$$s.t. \begin{cases} \sum_{i=1}^{n} x_{ij} \leqslant a_i, & \forall 1 \leqslant i \leqslant n \\ \sum_{j=1}^{m} x_{ij} = b_j, & \forall 1 \leqslant j \leqslant m \end{cases}$$

Можно заметить, что целевая функция суть $\mathrm{tr}(X(\alpha C + \beta T)^T)$, что позволяет упростить запись. $\alpha,\ \beta \in \mathbb{R}^+$ — некоторые весовые коэффициенты.

3 Это норма!

3.1 Что такое норма вектора? Что такое эквивалентность норм? Показать эквивалентность l_1, l_2 и l_∞ норм.

Определение. Здесь и далее через V — n-мерное линейное пространство над полем $\mathbb{F} \subseteq \mathbb{C}$.
Нормой называют отображение

$$\|\cdot\| : V \to \mathbb{R}^+$$

удовлетворяющее свойствам:

1. $\forall v \in V : \|v\| \geqslant 0 \wedge (\|v\| = 0 \iff v = 0_V)$
2. $\forall \alpha \in \mathbb{F}\ \forall v \in V : \|\alpha \cdot v\| = |\alpha| \cdot \|v\|$
3. $\forall u, v \in V : \|u + v\| \leqslant \|u\| + \|v\|$

Определение. Нормы $\|\cdot\|_\alpha : V \to \mathbb{R}^+,\ \|\cdot\|_\beta : V \to \mathbb{R}^+$ называются *эквивалентными*, если

$$\exists C_1, C_2 \in \mathbb{R}^+ : (0 \leqslant C_1 \leqslant C_2) \wedge (\forall v \in V : C_1\|v\|_\alpha \leqslant \|v\|_\beta \leqslant C_2\|v\|_\alpha)$$

Определение. $\forall x \in \mathbb{C}^n$:
$\|x\|_1 := \sum_{i=1}^{n} |x_i|$
$\|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$
$\|x\|_\infty := \max\limits_{1 \leqslant i \leqslant n} |x_i|$

Теорема 3.1. Нормы l_1, l_2 и l_∞ эквивалентны в $\mathbb{C}^n$.

Доказательство.

1. $l_1 \sim l_\infty : \|x\|_\infty = \max\limits_{1 \leqslant i \leqslant n} |x_i| \leqslant \sum_{i=1}^{n} |x_i| \leqslant n \max\limits_{1 \leqslant i \leqslant n} |x_i| = n\|x\|_\infty \implies C_1 = 1, C_2 = n$
2. $l_1 \sim l_2 : \|x\|_2 \leqslant \|x\|_1 \leqslant \sqrt{n}\|x\|_2$
   Левое неравенство доказывается возведением в квадрат обеих его частей, правое суть неравенство Коши-Буняковского для векторов $(1, \ldots, 1)$ и x.
3. $l_2 \sim l_\infty$ заключаем по транзитивности.

□

3.2 Что такое норма матрицы (линейного конечномерного оператора) порождённая векторной нормой? Выведите выражения для l_1, l_2 и l_∞ нормы матрицы. Что такое Фробениусова норма матрицы и какой аналог среди векторных норм она имеет?

Определение. $\forall A : V \to V$ — линейного конечномерного оператора

$$\|A\| := \sup\left\{\frac{\|Ax\|}{\|x\|} : x \in V \wedge x \neq 0\right\}$$

Теорема 3.2.

$\|A\|_1 := \max\limits_{1\leqslant j\leqslant n} \sum_{i=1}^n |a_{ij}|$ — наибольшая из l_1-норм столбцов матрицы оператора
$\|A\|_2 := |\sigma_{\max}(A)|$ — модуль наибольшего сингулярного значения оператора
$\|A\|_\infty = \max\limits_{1\leqslant i\leqslant n} \sum_{j=1}^n |a_{ij}|$ — наибольшая из l_1-норм строк матрицы оператора

Доказательство.

1. $\|A\|_1 := \max\limits_{1\leqslant j\leqslant n} \sum_{i=1}^n |a_{ij}|$.
   Пусть $\|x\|_1 = 1$, тогда

$$\begin{aligned}\|Ax\|_1 = \left\|\sum_{j=1}^n x_j \cdot A[:,j]\right\|_1 &\leqslant \sum_{j=1}^n \|x_j \cdot A[:,j]\|_1 \\ &= \sum_{j=1}^n |x_j| \cdot \|A[:,j]\|_1 \\ &\leqslant \|x\|_1 \max_{1\leqslant j\leqslant n} \|A[:,j]\|_1 = \max_{1\leqslant j\leqslant n} \|A[:,j]\|_1\end{aligned}$$

   где $A[:,j]$ и $A[i,:]$ имеют тот же смысл, что и в Python.

$$\|Ax\|_1 \leqslant \max_{1\leqslant j\leqslant n} \|A[:,j]\|_1$$

   Обратное неравенство получается подстановкой единичных ортов вместо вектора x в силу определения матричной нормы как супремума нормы образа оператора на единичной сфере.

2. $\|A\|_2 := |\sigma_{\max}(A)|$

$$\begin{aligned}\forall x : \|x\|_2 = 1 \implies & \\ \|Ax\|_2^2 &= (Ax, Ax) = (x, A^*Ax) \\ &\leqslant |\sigma_{\max}(A)|^2 \|x\|_2^2 = |\sigma_{\max}(A)|^2 \\ &\leqslant \sup_{\|x\|_2=1} \|Ax\|_2 = \|A\|_2\end{aligned}$$

   Сингулярные значения оператора A суть собственные значения эрмитова оператора A^*A. В силу эрмитовости у него его ОНСБ из собственных векторов. Разложив по нему вектор x и оценив коэффициенты сверху максимальным собственным значением как в прошлом пункте, получаем исходное утверждение.

3. $\|A\|_\infty = \max\limits_{1\leqslant i\leqslant n} \sum_{j=1}^n |a_{ij}|$ — тривиальное следствие определения l_∞-нормы.

□

Определение.

$$\|A\|_F := \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}$$

— Фробениусова норма оператора A. Она является естественным аналогом векторной l_2-нормы.

4 Вычисли это

4.1 Написать, что такое SVD и QR разложения матрицы A. Как можно использовать эти разложения при решении систем линейных уравнений и какова будет сложность решения?

Определение. Пусть $A \in \mathbb{C}^{m\times n}$, $\mathrm{rank}A = r$, тогда *SVD-разложением* матрицы A называют её представление в виде

$$A = U\Sigma V^*$$
$$U \in \mathbb{C}^{m\times r},\ U^*U = E_r,$$
$$\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_r),\ \sigma_1 \geqslant \sigma_2 \geqslant \dots \geqslant \sigma_r > 0$$
$$V \in \mathbb{C}^{n\times r},\ V^*V = E_r$$

Столбцы U называют левыми сингулярными векторами, столбцы матрицы V — правыми сингулярными векторами, числа σ_i — сингулярными значениями матрицы A. Имеет место эквивалентное выражение:

$$A = \sum_{i=1}^{r} \sigma_i U[:, i]V[:, i]^T$$

Определение. SVD-разложение матрицы $A = U\Sigma V^*$ позволяет определить т.н. *псевдообратную матрицу* или *обратную матрицу по Муру-Пенроузу*:

$$A^\dagger := V\Sigma^{-1}U^* \in \mathbb{C}^{n\times m} : A^\dagger A = E_n,\ AA^\dagger = E_m$$

В частный случаях полного столбцового и строчного ранга матрицы A также определяют односторонние псевдообразные матрицы:

$$\mathrm{rank}A = n \implies A^\dagger = (A^*A)^{-1}A^*$$
$$\mathrm{rank}A = m \implies A^\dagger = A^*(AA^*)^{-1}$$

Если A — квадратная невырожденная матрица, то её псевдообратная матрица равна обратной.

Утверждение 1. SVD-разложение позволяет решать СЛАУ

$$Ax = b,\ A \in \mathbb{C}^{m\times n},\ m \leqslant n$$

за время $O(mn^2)$.

Доказательство.

1. **Существование решения:**
   Частное решение имеет вид $\widehat{x} = A^\dagger b$ и является оптимальным в смысле МНК:
   $$\widehat{x} = \arg\min_{x\in\mathbb{C}^n} \|Ax - b\|_2$$

2. **Асимптотика вычисления решения:** Википедия предлагает следующий алгоритм:

   Numerical approach [edit]

   The SVD of a matrix **M** is typically computed by a two-step procedure. In the first step, the matrix is reduced to a bidiagonal matrix. This takes O(mn^2) floating-point operations (flop), assuming that $m \geq n$. The second step is to compute the SVD of the bidiagonal matrix. This step can only be done with an iterative method (as with eigenvalue algorithms). However, in practice it suffices to compute the SVD up to a certain precision, like the machine epsilon. If this precision is considered constant, then the second step takes O(n) iterations, each costing O(n) flops. Thus, the first step is more expensive, and the overall cost is O(mn^2) flops (Trefethen & Bau III 1997, Lecture 31).

Насколько я понял, за ним кроется следующая идея: свести задачу поиска SVD-разложения к задаче о ЖНФ. Сведение к двухдиагональной матрице, насколько я понимаю, упрощает вычисление сингулярных чисел, но не влияет на результат. Само сведение выглядит, например, так:

$$A^*A = (U\Sigma V^*)^*(U\Sigma V^*) = (V\Sigma U^*)(U\Sigma V^*) = V\Sigma^2 V^*$$
$$AA^* = \ldots = U\Sigma^2 U^*$$

А уже для вычисления собственных чисел и векторов есть очень разные алгоритмы, простейшим из которых будет метод Гаусса (медленный, численно неустойчивый, я всё это понимаю) как раз за $O(mn^2)$.

□

Определение. Пусть $M \in \mathbb{C}^{n\times p}$, $p \leqslant n$ имеет полный ранг p, тогда *QR-разложением* матрицы M называют её представление в виде

$$M = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}$$

где $Q_1 \in \mathbb{C}^{n\times p}$, $Q_2 \in \mathbb{C}^{n\times(n-p)}$, $R \in \mathbb{C}^{p\times p}$ — верхнетреугольная матрица с ненулевой главной диагональю, причём

$$Q_1^T Q = E_p,\ Q_2^T Q_2 = E_{n-p},\ Q_1^T Q_2 = O_{p\times(n-p)}$$

Утверждение 2. QR-разложение позволяет решать недоопределённые СЛАУ

$$Ax = b,\ A \in \mathbb{C}^{p\times n},\ p < n,\ \mathrm{rank}A = p$$

за $O(2p^2(n - p/3))$.

Доказательство.

1. **Существование решения:**
   Выпишем QR-разложение для A^T:

   $$A^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R$$

   Тогда докажем, что

   $$\widehat{x} := Q_1 R^{-T} b$$

   является решением исходной системы.

   $$A\widehat{x} = R^T Q_1^T Q_1 R^{-T} b = b$$

   В силу требования $Q_1^T Q_2 = 0$, получаем, что столбцы матрицы Q_2 образуют ФСР системы, т.е. общее решение имеет вид:

   $$\{x = \widehat{x} + Q_2 z \mid z \in \mathbb{C}^{n-p}\}$$

2. **Асимптотика вычисления решения:**
   Приведенную в формулировке оценку я взял из ученика Бойда, но пока не нашёл её доказательства. Тем не менее, кубическая верхняя граница времени вычисления QR -разложения ясна уже из тривиального алгоритма, основанного на ортогонализации по Граму-Шмидту. А именно, ортонормируем систему столбцов матрицы A и составим из полученных векторов матрицу Q. Проекции, вычитавшиеся на каждом шаге алгоритма, запишем в матрицу R с противоположным знаком. По построению понятно, что полученные матрицы подходят

под определение QR-разложения. Если не возражаете, вместо того, чтобы записывать это, я вставлю скриншот:

1 Gram-Schmidt process

Consider the GramSchmidt procedure, with the vectors to be considered in the process as columns of the matrix A. That is,

$$A = \left[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n \right].$$

Then,

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{a}_1, \qquad \mathbf{e}_1 = \frac{\mathbf{u}_1}{||\mathbf{u}_1||}, \\ \mathbf{u}_2 &= \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\mathbf{e}_1, \qquad \mathbf{e}_2 = \frac{\mathbf{u}_2}{||\mathbf{u}_2||}. \\ \mathbf{u}_{k+1} &= \mathbf{a}_{k+1} - (\mathbf{a}_{k+1} \cdot \mathbf{e}_1)\mathbf{e}_1 - \cdots - (\mathbf{a}_{k+1} \cdot \mathbf{e}_k)\mathbf{e}_k, \qquad \mathbf{e}_{k+1} = \frac{\mathbf{u}_{k+1}}{||\mathbf{u}_{k+1}||}. \end{aligned}$$

Note that $||\cdot||$ is the L_2 norm.

1.1 QR Factorization

The resulting QR factorization is

$$A = \left[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n \right] = \left[\mathbf{e}_1 \mid \mathbf{e}_2 \mid \cdots \mid \mathbf{e}_n \right] \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{e}_1 & \mathbf{a}_2 \cdot \mathbf{e}_1 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_1 \\ 0 & \mathbf{a}_2 \cdot \mathbf{e}_2 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_n \end{bmatrix} = QR.$$

Алгоритм Грама-Шмидта совершает r итераций (т.к. полный столбцовый ранг), на j-ой вычисляет $j-1$ проекций, затрачивая $O(n)$ операций на каждую. Итого получаем что-то в духе $O\left(\frac{nr(r-1)}{2}\right)$, что по порядку величин совпадает с заявленной оценкой.

□

4.2 Что такое разреженная матрица? Какие существуют форматы хранения таких матриц в памяти, опишите их структуру и способ доступа к элементу матрицы? Какие преимущества даёт работа с разреженными матрицами при вычислениях?

Определение. Матрица называется *разреженной*, если доля нулей среди её элементов превышает некоторый порог (подозреваю, что он зависит от того, какую природу имеет эта матрица).

Давайте рассмотрим форматы хранения разреженных матриц в памяти компьютера на примере библиотеки SciPy. В ней предлагаются семь вариантов, но я выбрал из них только основные, т.к. остальные являются их разного рода модификациями:

- **coo_matrix** — **COO**rdinate format matrix — три numpy-массива, хранящих ненулевых элементы матрицы: row, column, data. Самый естественный формат, он же первый, который приходит на ум. Из недостатков можно отметить, в первую очередь, невозможность непосредственного выполнения арифметических операций над матрицами, закодированными в таком виде. Для доступа к элементу матрицы нужно сначала найти его индекс. Это легко делать за $O(\log n)$ бинпоиском, если массив координат предварительно отсортировать.
- **csc_matrix** — **C**ompressed **S**parse **C**olumn matrix — три numpy-массива, оптимизированные для столбцовых операций с матрицей: indptr, indices, data.
  - indices[indptr[i] : indptr[i+1]] — индексы строк ненулевых элементов i-го столбца.

– data[indptr[i] : indptr[i+1]] — сами эти элементы.

– Доступ к $a_{ij} \neq 0$: data[indptr[j] + k], где k — позиция i в indices[indptr[j] : indptr[j+1]]

Такой формат уже допускает эффективную реализацию арифметики (и большинство пакетов его поддерживают), но операции со строками матрицы в нём, по очевидным причинам, медленные.

- **csr_matrix** — **C**ompressed **S**parse **R**ow matrix — ccr_matrix, только для строк.
- **dia_matrix** — Sparse matrix with **DIA**gonal storage — специальный формат, нужный, насколько я понял, для решения ДУРЧП. В нём просто хранятся все диагонали матрицы, на которых есть хотя бы один ненулевой элемент, в виде numpy-массивов длиной в главную диагональ исходной матрицы. Кроме них также хранится массив свдигов, который сопоставляет массивам относительное положение хранящейся в них диагонали относительно главной.
  Доступ к элементу в таком формате реализуется тривиально.
  Очевидным недостатком является необходимость хранить диагональ целиком (ради одного ненулевого элемента он может хранить миллионы нулей), так что использовать DIA без уважительной причины — моветон.
- **dok_matrix** — **D**ictionary **O**f **K**eys based sparse matrix — coo_matrix со словарём вместо списка координат. Он проще устроен, позволяет доступ к элементам за $O(1)$, но время работы арифметических операций сильно зависит от эффективности реализации словаря. Тем не менее, это подходящий формат для тех ситуаций, когда про данные непонятно ничего, кроме того, что они разреженные, особенно если их нужно только хранить и читать.
- **lil_matrix** — Row-based **LI**nked **L**ist sparse matrix — хранит numpy-массив списков ненулевых элементов строк и их столбцовых индексов. Просто, удобно, но арифметические операции тоже работают медленно, а работать со столбцами и вовсе очень плохая идея.

Поскольку в реальных задачах данные чаще всего разреженные (характерный пример — матрица инцидентности графа друзей в Вк), хранение их в одном из приведенных выше форматов не прихоть, а необходимость. К тому же, грамотный выбор формата позволяет не тратить уйму процессорного времени на бессмысленные операции с нулями.

5 Диаграмма Вороного

Пусть $x_1, \ldots, x_k \in \mathbb{R}^n$. Рассмотрим множество точек, которые ближе к x_0, чем к $x_1, \ldots, x_k$:

$$V = \{x \in \mathbb{R}^n \mid \|x - x_0\|_2 \leqslant \|x - x_i\|_2,\ i = 1, \ldots, k\}$$

Множество V называется *областью Вороного* для точки x_0.

5.1 Докажите, что множество V является многоугольником. Представьте его в виде $\{x \in \mathbb{R}^n \mid Ax \preceq b\}$. Обозначение $x \preceq y$ означает, что $x_i \leqslant y_i,\ i = 1, \ldots, n$.

Проведём отрезок, соединяющий x_0 и некоторую x_i. Проведём перпендикулярно ему гиперплоскость, проходящую через середину отрезка. Все точки, которые лежат в одной полуплоскости с x_0, ближе к x_0, чем к x_i. Область Вороного для точки x_0 будет пересечением k таких полуплоскостей, т.е. многоугольником (возможно неограниченным).

Зададим этот многоугольник аналитически:

$$\begin{aligned}\|x - x_0\|_2^2 \leqslant \|x - x_i\|_2^2 \\ &\iff (x - x_0)^T(x - x_0) \leqslant (x - x_i)^T(x - x_i) \\ &\iff x^T x - 2x_0^T x + x_0^T x_0 \leqslant x^T x - 2x_i^T x + x_i^T x_i \\ &\iff 2(x_i - x_0)^T x \leqslant x_i^T x_i - x_0^T x_0\end{aligned}$$

Получили систему линейных неравенств:

$$\begin{bmatrix} a_1^T x \\ \dots \\ a_k^T x \end{bmatrix} := \begin{bmatrix} (x_1 - x_0)^T x \\ \dots \\ (x_k - x_0)^T x \end{bmatrix} \leqslant \begin{bmatrix} x_1^T x_1 - x_0^T x_0 \\ \dots \\ x_k^T x_k - x_0^T x_0 \end{bmatrix} =: \begin{bmatrix} b_1 \\ \dots \\ b_k \end{bmatrix}$$

Итого получаем искомое представление $Ax \preceq b$.

5.2 Обратно, покажите как по данному многоугольнику восстановить точки $x_0, \dots, x_k$, для которых он определяет область Вороного.

Выберем произвольную точку, лежащую в области, и рассмотрим её образы при отражении относительно граничных гиперплоскостей — обозначим их как x_i для i-ой стороны многоугольника. По построению, рассуждения предыдущего пункта дают в точности ту же область Вороного Приведенное рассуждение заодно показывает, что между областями Вороного и точками $x_0, x_1, \dots, x_k$ нет взаимно-однозначного соответствия.

5.3 Сделайте обобщение областей Вороного для $n > 1$ точек и покажите, как по данному разбиению пространства на многоугольники (возможно открытые) восстановить множество точек, для которого это разбиение является разбиением Вороного.

Достаточно построить области Вороного для каждой из точек в отдельности, этого хватит для того, чтобы задать разбиение пространства (для каждой точки из $\mathbb{R}^n$ есть ближайшая из данного набора; точки не могут принадлежать одновременно внутренности двух областей).

Утверждение 3. Разбиение пространства $\mathbb{R}^n$ на k многоугольников $P_i := \{x \in \mathbb{R}^n \mid A_i x \preceq b_i\}$ является диаграммой Вороного тогда и только тогда, когда существует k точек $x_1, \dots, x_k$, т.ч. $x_i \in P_i$ и для всякой пары смежных многоугольников P_i и P_j x_i получается из x_j отражением относительно общей стороны. Т.к. отражение относительно гиперплоскости — это аффинное преобразование, а многоугольники заданы линейными неравенствами, вопрос сводится к поиску решения задачи линейного программирования. Найденные точки, удовлетворяющие всем линейным ограничениям, и будут порождающими, если таковые имеются.

5.4 Выберите область науки и технологии, которая Вам интересна и опишите, как в ней можно использовать разбиение Вороного. Для вдохновения можете посмотреть статью в Википедии

Мне понравилось несколько коротких сюжетов из разных областей технологии, красной нитью через которые проходит диаграмма Вороного. Все сюжеты объединяет общая задача: расположить некоторый объект в замкнутой области пространства на максимальном удалении от уже имеющихся. К примеру, построить саркофаг для токсичных отходов вдали от городов, поставить радиовышку в городе наиболее целесообразным способом, спланировать маршрут, максимально удалённый от радаров (в целях стратегического шпионажа). Ответ всегда лежит на рёбрах и в узлах диаграммы Вороного, что очевидно из определения.
Т.к. я очень люблю музыку, мне также закономерно пришла в голове идея рекомендательного

музыкального сервиса. Он бы не взлетел, т.к. требует слишком много действий от пользователя, но всё же. Можно просить классифицировать настроение песни (количество категорий зависит исключительно от фантазии разработчика) и брать процент положительных голосов в качестве координаты. Для тех аудиозаписей, для которых известен жанр, эти голоса бы учитывались и в статистике этого жанра. Когда наберётся достаточно данных, на жанрах как на числовых векторах можно построить диаграмму Вороного. Далее можно было бы реализовать поиск музыки по настроению: пользователь выставляет желаемый уровень каждой эмоции (или переносит его с интересующего его трека) и получает наиболее подходящий жанр. Я не утверждаю, что такой метод лучше тех, что применяются обычно, но о таком применении диаграммы Вороного я раньше не думал.

6 Выпуклый или конический

Определение. Множество $X \subseteq \mathbb{R}^n$ называется *выпуклым*, если

$$\forall x, y \in X \ \forall \alpha \in [0,1] : \alpha x + (1-\alpha) y \in X$$

Определение. Множество $X \subseteq \mathbb{R}^n$ называется *коническим*, если

$$\forall x \in X \ \forall \alpha \in \mathbb{R}^+ : \alpha x \in X$$

Утверждение 4. Полуплоскость $a^T x \leqslant b$ является коническим множеством тогда и только тогда, когда её граница содержит начало координат.

Доказательство. И это более-менее очевидно из геометрических соображений. □

Задача 1. Определить, являются ли следующие множества выпуклыми и/или коническими.

1. $X := \{x \in \mathbb{R}^n \mid \alpha \leqslant a^T x \leqslant \beta\}$
2. $X := \{x \in \mathbb{R}^n \mid a_1^T x \leqslant b_1,\ a_2^T x \leqslant b_2\}$
3. $X := \{x \mid \|x - x_0\|_2 \leqslant \|x - y\|_2,\ \forall y \in S \subseteq \mathbb{R}^n\}$
4. Множество точек X, расстояние от которых до $\mathbf{a}$ не превышает доли $\theta \in [0,1]$ от расстояния до точки $\mathbf{b}$: $\{x \mid \|x - a\|_2 \leqslant \theta \|x - b\|_2\}, a \neq b$.

Доказательство.

1. $X := \{x \in \mathbb{R}^n \mid \alpha \leqslant a^T x \leqslant \beta\}$
   Пусть $x, y \in X$, $\gamma \in [0,1]$, тогда:

   $$a^T(\gamma x + (1-\gamma)y) = \gamma a^T x + (1-\gamma) a^T y \leqslant \gamma\beta + (1-\gamma)\beta \leqslant \beta$$

   Оценка снизу получается аналогичным образом. Т.е. множества вида $a^T x \leqslant b, a^T x \geqslant b$ выпуклые, а класс выпуклых множеств замкнут относительно пересечения. Тем не менее, коническим оно не является, т.к. для заданного $x \in X$ при $\gamma \geqslant \max\{|\alpha|, |\beta|\} : \gamma x \notin X$.
   Ответ: выпуклое; коническое только при $\alpha = \beta = 0$.

2. $X := \{x \in \mathbb{R}^n \mid a_1^T x \leqslant b_1,\ a_2^T x \leqslant b_2\}$
   Является выпуклым как пересечение выпуклых множеств (полуплоскостей). Не обязательно является коническим, контрпример — $\{x \in \mathbb{R}^2 \mid x + y \leqslant -1, x + 2y \leqslant -2\}$. Приведенное множество не содержит начало координат, а потому не может содержать луч, соединяющий принадлежащую ему точку и $(0,0)$.
   Ответ: выпуклое; вообще говоря, не коническое.

3. $X := \{x \mid \|x - x_0\|_2 \leqslant \|x - y\|_2,\ \forall y \in S \subseteq \mathbb{R}^n\}$
   Проведём из x_0 перпендикуляр в каждую точку $y \in S$ и проведём через середину такого перпендикуляра гиперплоскость. Выберем далее ту полуплоскость, которая содержит точку x_0. Пересечение любого числа выпуклых множеств является выпуклым множеством, потому X — выпуклое множество.
   Коническим оно, тем не менее, может и не являться. Возьмём в качестве S сферу с центром в точке x_0. Тогда X будет сферой половинного радиуса с тем же центром, а ограниченное невырожденное множество конусом не является.
   Ответ: выпуклое; вообще говоря, не коническое

4. Множество точек X, расстояние от которых до $\mathbf{a}$ не превышает доли $\theta \in [0, 1]$ от расстояния до точки $\mathbf{b}$: $\{x \mid \|x - a\|_2 \leqslant \theta\|x - b\|_2\}, a \neq b$.
   Пусть $\theta = 0$, тогда $X = \{a\}$ и множество X выпукло, а при $a = 0$ — ещё и коническое.
   Пусть $\theta = 1$, тогда X — полуплоскость, граничная гиперплоскость которой перпендикулярна отрезку, соединяющему a с b, и проходит через его середину. Пусть $\theta \in (0, 1)$. Тогда возведём в квадрат обе части неравенства:

$$\|x - a\|_2^2 \leqslant \theta^2\|x - b\|_2^2$$

   Приведением подобных получаем n-мерный шар (все квадратичные слагаемые войдут в уравнение с коэффициентами $(1 - \theta) > 0$), который является выпуклым множеством (неравенство треугольника), но не является конусом (хотя бы потому, что это ограниченное множество).
   Ответ:

   (a) $\theta = 0$: выпуклое; коническое при $a = 0$ (одна точка)

   (b) $\theta = 1$: выпуклое; коническое, если граница содержит начало координат (полуплоскость)

   (c) $\theta \in (0, 1)$: выпуклое; не коническое (n-мерный шар)

□