Симплекс

Алиса Саютина

29 июля 2019

Содержание

1. Симплекс

Note: это старый материал, 2019 года. Повествование можно улучшить, если воспользоваться больше алгебраическими свойствами и меньше прямыми соображениям, но сейчас у автора нет на это времени.

Автор благодарит Серёже Копелиовичу

1.1. Линейное программирование

Определим задачу линейного программирования следующим образом:

$$\begin{cases} a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 + \ldots + a_{1,n}x_n \leqslant b_1 \\ a_{2,1} \cdot x_1 + a_{2,2} \cdot x_2 + \ldots + a_{2,n}x_n \leqslant b_2 \\ \ldots \\ a_{m,1} \cdot x_1 + a_{m,2} \cdot x_2 + \ldots + a_{m,n}x_n \leqslant b_m \\ x_1 \geqslant 0 \\ x_2 \geqslant 0 \\ \ldots \\ x_n \geqslant 0 \end{cases}$$

Или, если коротко $Ax \leqslant b$, $x \geqslant 0$.

Другие формы задач линейного программирования:

1. $Ax \leqslant b$
2. $Ax \leqslant b$, $x \geqslant 0$
3. $Ax = b$, $x \geqslant 0$

Заметим, что все формы эквивалентны друг другу.

- (2) $\to$ (1): просто добавим уравнения $-x_i \leqslant 0$ для всех i.
- (1) $\to$ (2): заменим переменную x_i на $u_i - v_i$, где $u_i \geqslant 0$, $v_i \geqslant 0$
- (2) $\to$ (3): пусть уравнение номер t это $a_1x_1 + a_2x_2 + \ldots + a_nx_n \leqslant b$.
  Заменим его на $a_1x_1 + a_2x_2 + \ldots + a_nx_n + \mathbf{y_t} = b$
- (3) $\to$ (2): $Ax = b \iff Ax \leqslant b$ и $(-A)x \leqslant b$.
  на практике такой замены следует избегать, так как она приводит к увеличению системы в два раза.

Также часто требуется найти не *какое-то* решение, а решение оптимизирующее некоторую линейную функцию: $c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max$, или $\langle c, x\rangle \to \max$.

В целом это несущественное дополнение, так как можно было бы добавить уравнение $\langle c, x\rangle \geqslant ans$ и сделать бинпоиск по ответу.

1.2. Нормальная форма и основы симплекса

Предположим, что нам дана система уравнений в *нормальной* форме.

***Определение* 1.1.** Пусть n это число переменных, а m это количество уравнений.

Нормальной назовём форму $Ax = b$ с дополнительными ограничениями:

- $A = (E_m|A')$
- $\forall i\colon b_i \geqslant 0$
- $\forall i \in \{1, 2, \ldots, m\}\colon c_i = 0$.

Например, такая система может выглядеть так:

Пример.

$$\begin{cases} 1 \cdot x_1 + 0 \cdot x_2 + x_3 - x_4 = 5 \\ 0 \cdot x_1 + 1 \cdot x_2 + 2 \cdot x_3 + x_4 = 3 \\ 0 \cdot x_1 + 0 \cdot x_2 + x_3 + x_4 \to \max \end{cases}$$

***Определение* 1.2.** Сразу назовём переменные $x_1, x_2, \ldots, x_m$ *базисными*.

Заметим, что такая форма записи не совсем изоморфна предыдущим: например, в такой форме у любой системы есть хоть какое-то решение (например, $x_i = b_i$ для $1 \leqslant i \leqslant m$ и $x_i = 0$ иначе), что неверно для предыдущих форм.

Подробности по переводу в нормальную форму обсудим несколько позже, а пока приступим к решению задачию

Лемма. Заметим, что если все $c_i \leqslant 0$, то текущее решение $x_i = b_i$ оптимально.

Иначе, существует какое-то i, что $c_i > 0$. Сейчас переменная $x_i = 0$, попробуем её увеличить. Что нам может помешать? Если $a_{k,i} \neq 0$, то начнёт меняться переменная x_k. Если она меняется в сторону увеличения, то ничего плохого не произойдёт. Но если она уменьшается, то надо сделать так, чтобы она осталась неотрицательной.

Предполагая, что остальные переменные не меняются, имеем уравнения: $x_k + a_{k,i}x_i = b_k$ (для $1 \leqslant k \leqslant m$).

Тем самым максимальное увеличение x_i равно

$$\min_{\substack{1 \leqslant k \leqslant m \\ a_{k,i} > 0}} b_k/a_{k,i}$$

Если таких k нет, то имеет случай бесконечного значения функции $\langle c, x\rangle$.

Иначе хочется увеличить x_i и уменьшить базисные переменные, но хочется сохранить нормальность постановки. Для этого достаточно проделать следующие действия (пусть k = `argmin` формулы выше):

- Нормализация: поделим все коэффициенты в строке k (включая b_k) на $a_{k,i}$.
- Вычитание: вычтем строку k из всех остальных равенств с нужным коэффициентами чтобы занулить столбец i.
- Вычитание (2): вычтем строку k также из строки c.

  Заметим, что это преобразование корректно. До этого мы оптимизировали $f(\vec{x}) \to \max$, а теперь оптимизируем $f(\vec{x}) - \text{const} \to \max$.
- Замена базиса: заметим, что переменная x_k перестала удовлетворять требованию на нормальность, однако x_i теперь ей удивляет. Можно переставить столбцы k и i чтобы снова получить нормальную матрицу.

Замечание. (про реализацию)

- Как и в Гауссе, удобно хранить коэффициент b_t вместе со всей строкой $a_{t,*}$
- Удобно хранить строку c в той же матрице, что и матрица $(A|b)$, тогда второй и третий шаг выше превращаются в один.
- При этом клеточка строки c под коэффициентами b тоже оказывается полезной: со временем, туда со знаком $(-)$ попадут все const выше, а значит там записан $-\text{ans}$.
- При замене базиса можно не свопать столбцы физически, а просто поддерживать для каждой строки какая переменная является базисной для неё.

Назовём операцию замены базиса (столбец i по строке k) как `pivot(i, k)`.

1.3. Симплекс-метод

Хочется сказать, что надо просто проделывать операцию замены базиса до тех пор, пока мы не достигнем *критерия остановки* (все c_i во внебазисных переменных равны 0).

В таком случае асимптотика составляет $\mathcal{O}(nm \cdot k)$, где k число шагов симплекса. Вопрос чему равно k.

Однако, может так получится, что базис мы заменили, а вычисленный лимит равен нулю. Тогда базис заменили, а толку мало. В некоторых реализациях может вообще получиться, что алгоритм **зацикливается**.

Лемма. В предположении, что симплекс не зацикливается, он отрабатывает за $\leqslant C_n^m$ итераций.

Доказательство. В каждый момент в алгоритме симплекса есть какие-то переменные, которые не входят в базис (равны 0), и остальные.

Всего есть не более $C_n^{n-m} = C_n^m$ способов выбрать переменные не в базисе. Если зафиксировать переменные не в базисе, то остальные вычисляются однозначно, это кандидат на ответ. □

Утверждение 1.1. (Правило Блэнда) Если каждый раз выбирать минимальное j, что $c_j > 0$, то наборы базисов в алгоритме не будет повторяться.

Доказательство. Без доказательства □

Итого худшее время работы конечное, хотя и возможно экспоненциально большое. Существуют известные тесты с экспоненциальным временем работы. На более-менее не специфичных данных можно ожидать что $k = \mathcal{O}(n + m)$.

1.4. Нормализация задачи

Теперь обсудим как перейти к нормальной форме задачи.

Если исходная форма задачи выглядит как $Ax \leqslant b$, то можно просто воспользоваться предложенным выше способом:

$Ax \leqslant b$ превращается в $Ax + y = b$.

Приятным бонусом также является автоматическое наличие базисных столбцов: это собственно переменные $y_1, y_2, \ldots, y_m$.

Если задача дана в форме $Ax = b$, то можно было бы записать $Ax \leqslant b$, $(-A)x \leqslant -b$ и повторить всё сказанное выше, но это раздувает систему более чем в два раза, что плоховато.

Эффективное решение это запустить алгоритм Гаусса для системы $Ax = b$. Гаусс приведёт часть матрицы к диагональном виду, что нам и нужно. В результате работы Гаусса могут образоваться уравнения вида $0 \cdot x = 0$ которое следует удалить, или уравнения $0 \cdot x = a$ $(a \neq 0)$, в таком случае решения нет.

1.5. Поиск начального решения

Тем не менее, осталось ещё одно требование нормальности, которое мы не выполнили: $b_i \geqslant 0$.

Пусть у нас есть задача $Ax = b$, где A и b удовлетворяют требованиям нормальности.

Давайте в каждое уравнение добавим фиктивную переменную t (общую на всех): $Ax - t = b$.

Утверждение 1.2. Новая система **всегда** имеет решение относительно (x, t).

Доказательство. Выберем $t = -\min b_i$.

Тогда $Ax = t + b$ — задача в нормальной форме с правой частью $\geqslant 0$, мы знаем стандартный способ получить хотя бы какое-то решение в таком случае. □

Утверждение 1.3. Будем решать задачу $Ax - t = b$, оптимизируя $-t \to \max$.

Если получилось что $t > 0$, то решения исходной системы не существует.

Доказательство. Допустим что исходная система имеет решение x.

Тогда $(x, 0)$ решение новой системы. □

Хотим запустить симплекс на задаче

$$\begin{cases} Ax - t = b \\ -t \to \max \end{cases}$$

К сожалению эта система хотя и имеет решение, она не является нормальной.

Пусть $k = \texttt{argmin}\ b_k$, а $n + 1$ — номер переменной t.

Тогда не сложно видеть, что `pivot(n + 1, k)` избавит нашу систему (с фиктивной переменной) от отрицательных b_i. Разберём этот факт на примере (переменные x_1 и x_2 изначально являются базисными):

$$\begin{cases} 1 \cdot x_1 + 0 \cdot x_2 + x_3 - x_4 - t = -3 \\ 0 \cdot x_1 + 1 \cdot x_2 - x_3 + 2 \cdot x_4 - t = -4 \\ -t \to \max \end{cases}$$

Что будет происходить? Минимальным b_k является b_2. Значит первым делом мы отнормируем вторую строку:

$$\begin{cases} 1 \cdot x_1 + 0 \cdot x_2 + x_3 - x_4 - t = -3 \\ 0 \cdot x_1 - 1 \cdot x_2 + x_3 - 2 \cdot x_4 + t = 4 \\ -t \to \max \end{cases}$$

А потом добавим эту строку ко всем остальным, чтобы больше не было вхождений t:

$$\begin{cases} 1 \cdot x_1 - 1 \cdot x_2 + 2 \cdot x_3 - 3 \cdot x_4 = 1 \\ 0 \cdot x_1 - 1 \cdot x_2 + x_3 - 2 \cdot x_4 + t = 4 \\ 0 \cdot x_1 - 1 \cdot x_2 + x_3 - 2 \cdot x_4 + 4 \to \max \end{cases}$$

Мы получили корректную нормальную систему: в правой части записаны числа $\geqslant 0$, базисом является (x_1, t).

После чего мы применяем метод симплекса и решаем полученную систему. Если получилось что $t > 0$, то решения по утверждению выше нет.

Иначе (так как $t \geqslant 0$), то $t = 0$. Теперь нужно просто аккуратно удалить переменную t назад из системы и перейти к решению исходной системы.

- Переменная t **не лежит** в базисе.

  Тогда можно просто выкинуть (или просто занулить) это столбец и вернутся к решению исходной задачи оптимизации.

- Переменная t **лежит** в базисе, пусть она базисная для k-й строки.

  Тогда нужно найти любой другой столбец i, который не лежит в базисе и имеет ненулевой коэффициент в k-й строке и произвести операцию `pivot(i, k)`.

  После чего удалим переменную t как было описано выше.

Остался единственный нюанс: может показаться что алгоритм выглядит так:

- Заменили для системы исходный вектор оптимизации c на новый вектор оптимизации c' (соответствующий $-t \to \max$),
- Сделали `pivot` чтобы получить нормализацию,
- `(*)` Решили $-t \to \max$,
- Удалили переменную t,
- `(**)` Подцепили назад вектор c и стали решать старую задачу.

Это не совсем правда, потому что преобразования в шаге `(*)` должны были влиять и на c. Поэтому стоит подцепить в систему сразу два вектора оптимизации: основной и «запасной».

Принимать решения какой столбец pivot-ить будем по основному, но пересчитывать сразу оба.

Изначально основной это $-t \to \max$ а запасной это исходный вектор c. Но перед шагом `(**)` мы их меняем местами.