Симплекс-метод

Симплекс-метод

Текущая версия (не проверялась)

Не путать с «симплекс-методом» — методом оптимизации произвольной функции. См. Метод Нелдера — Мида

Симплекс-метод — алгоритм решения оптимизационной задачи линейного программирования путём перебора вершин выпуклого многогранника в многомерном пространстве. Метод был разработан американским математиком Джорджем Данцигом (George Dantzig) в 1947 году.

Содержание

1 Описание

2 Алгоритм симплекс-метода

2.1 Усиленная постановка задачи

2.2 Алгоритм

3 Двухфазный симплекс-метод

3.1 Причины использования

3.2 Модификация ограничений

3.2.1 Различия между дополнительными и вспомогательными переменными

3.3 Фазы решения

4 Модифицированный симплекс-метод

5 Двойственный симплекс-метод

6 Литература

[править] Описание

Переход от одной вершины к другой

Задача линейного программирования состоит в том, что необходимо максимизировать или минимизировать некоторый линейный функционал на многомерном пространстве при заданных линейных ограничениях.

Заметим, что каждое из линейных неравенств на переменные ограничивает полупространство в соответствующем линейном пространстве. В результате все неравенства ограничивают некоторый многогранник (возможно, бесконечный), называемый также полиэдральным конусом. Уравнение W(x) = c, где W(x) — максимизируемый (или минимизируемый) линейный функционал, порождает гиперплоскость L(c). Зависимость от c порождает семейство параллельных гиперплоскостей. Тогда экстремальная задача приобретает следующую формулировку — требуется найти такое наибольшее c, что гиперплоскость L(c) пересекает многогранник хотя бы в одной точке. Заметим, что пересечение оптимальной гиперплоскости и многогранника будет содержать хотя бы одну вершину, причём, их будет более одной, если пересечение содержит ребро или k-мерную грань. Поэтому максимум функционала можно искать в вершинах многогранника. Принцип симплекс-метода состоит в том, что выбирается одна из вершин многогранника, после чего начинается движение по его рёбрам от вершины к вершине в сторону увеличения значения функционала. Когда переход по ребру из текущей вершины в другую вершину с более высоким значением функционала невозможен, считается, что оптимальное значение c найдено.

Последовательность вычислений симплекс-методом можно разделить на две основные фазы:

нахождение исходной вершины множества допустимых решений,

последовательный переход от одной вершины к другой, ведущий к оптимизации значения целевой функции.

При этом в некоторых случаях исходное решение очевидно или его определение не требует сложных вычислений, например, когда все ограничения представлены неравенствами вида «меньше или равно» (тогда нулевой вектор совершенно точно является допустимым решением, хотя и, скорее всего, далеко не самым оптимальным). В таких задачах первую фазу симплекс-метода можно вообще не проводить. Симплекс-метод, соответственно, делится на однофазный и двухфазный.

[править] Алгоритм симплекс-метода

[править] Усиленная постановка задачи

Рассмотрим следующую задачу линейного программирования:

Теперь поставим эту задачу в эквивалентной усиленной форме. Необходимо максимизировать Z, где:

Здесь x — переменные из исходного линейного функционала, xs — новые переменные, дополняющие старые таким образом, что неравенство переходит в равенство, c — коэффициенты исходного линейного функционала, Z — переменная, которую необходимо максимизировать. Полупространства и в пересечении образуют многогранник, представляющий множество допустимых решений. Разница между числом переменных и уравнений даёт нам число степеней свободы. Проще говоря, если мы рассматриваем вершину многогранника, то это число рёбер, по которым мы можем продолжать движение. Тогда мы можем присвоить этому числу переменных значение 0 и назвать их «непростыми». Остальные переменные при этом будут вычисляться однозначно и называться «простыми». Полученная точка будет вершиной в пересечении соответствующих непростым переменным гиперплоскостей. Для того, чтобы найти т. н. начальное допустимое решение (вершину, из которой мы начнём движение), присвоим всем изначальным переменным x значение 0 и будем их считать непростыми, а все новые будем считать простыми. При этом начальное допустимое решение вычисляется однозначно : .

[править] Алгоритм

Теперь приведём шаги алгоритма. На каждом шаге мы будем менять множества простых и непростых векторов (двигаться по рёбрам), и матрица будет иметь следующий вид:

где cB — коэффициенты вектора c соответствующие простым переменным (переменным xs соответствуют 0), B — столбцы , соответствующие простым переменным. Матрицу, образованную оставшимися столбцами обозначим D. Почему матрица будет иметь такой вид поясним в описании шагов алгоритма.

Первый шаг.

Выбираем начальное допустимое значение, как указано выше. На первом шаге B — единичная матрица, так как простыми переменными являются xs. cB — нулевой вектор по тем же причинам.

Второй шаг

Покажем, что в выражении только непростые переменные имеют ненулевой коэффициент. Заметим, что из выражения Ax+xs=b простые переменные однозначно выражаются через непростые, так как число простых переменных равно числу уравнений. Пусть x ' — простые, а x ' ' — непростые переменные на данной итерации. Уравнение Ax+xs=b можно переписать, как Bx '+Dx ' '=b. Умножим его на B − 1 слева: x' + B − 1Dx'' = B − 1b. Таким образом мы выразили простые переменные через непростые, и в выражении B − 1Ax + B − 1xs, эквивалентному левой части равенства, все простые переменные имеют единичные коэффициенты. Поэтому, если прибавить к равенству Z − cTx = 0 равенство , то в полученном равенстве все простые переменные будут иметь нулевой коэффициент — все простые переменные вида x сократятся, а простые переменные вида xs не войдут в выражение .

Выберем ребро, по которому мы будем перемещаться. Поскольку мы хотим максимизировать Z, то необходимо выбрать переменную, которая будет более всех уменьшать выражение

.

Для этого выберем переменную, которая имеет наибольший по модулю отрицательный коэффициент. Если таких переменных нет, то есть все коэффициенты этого выражения неотрицательны, то мы пришли в искомую вершину и нашли оптимальное решение. В противном случае начнём увеличивать эту непростую переменную, то есть перемещаться по соответствующему ей ребру. Эту переменную назовём входящей.

Третий шаг

Теперь необходимо понять, какая простая переменная первой обратится в ноль по мере увеличения входящей переменной. Для этого достаточно рассмотреть систему:

При фиксированных значениях непростых переменных система однозначно разрешима относительно простых, поэтому мы можем определить, какая из простых переменных первой достигнет нуля при увеличении входящей. Эту переменную назовем выходящей. Это будет означать, что мы натолкнулись на новую вершину. Теперь входящую и выходящую переменную поменяем местами — входящая «войдёт» в простую, а выходящая из них «выйдет» в непростые. Теперь перепишем матрицу B и вектор cB в соответствии с новыми наборами простых и непростых переменных, после чего вернёмся ко второму шагу. x''

Поскольку число вершин конечно, то алгоритм однажды закончится. Найденная вершина будет являться оптимальным решением.

[править] Двухфазный симплекс-метод

[править] Причины использования

Если в условии задачи линейного программирования не все ограничения представлены неравенствами типа «≤», то далеко не всегда нулевой вектор будет допустимым решением. Однако каждая итерация симплекс-метода является переходом от одной вершины к другой, и если неизвестно ни одной вершины, алгоритм вообще не может быть начат.

Процесс нахождения исходной вершины не сильно отличается от однофазного симплекс-метода, однако может в итоге оказаться сложнее, чем дальнейшая оптимизация.

[править] Модификация ограничений

Все ограничения задачи модифицируются согласно следующим правилам:

ограничения типа «≤» переводятся на равенства созданием дополнительной переменной с коэффициентом «+1». Эта модификация проводится и в однофазном симплекс-методе, дополнительные переменные в дальнейшем используются как исходный базис.

ограничения типа «≥» дополняются одной переменной с коэффициентом «−1». Поскольку такая переменная из-за отрицательного коэффициента не может быть использована в исходном базисе, необходимо создать ещё одну, вспомогательную, переменную. Вспомогательные переменные всегда создаются с коэффициентом «+1».

ограничения типа «=» дополняются одной вспомогательной переменной.

Соответственно, будет создано некоторое количество дополнительных и вспомогательных переменных. В исходный базис выбираются дополнительные переменные с коэффициентом «+1» и все вспомогательные. Осторожно: решение, которому соответствует этот базис, не является допустимым.

[править] Различия между дополнительными и вспомогательными переменными

Несмотря на то, что и дополнительные, и вспомогательные переменные создаются искусственно и используются для создания исходного базиса, их значения в решении сильно отличаются:

дополнительные переменные сообщают, насколько соответствующее им ограничение «недоиспользовано». Значение дополнительной переменной нулю соответствует равенству значений правых и левых частей ограничения.

вспомогательные переменные сообщают, насколько данное условие далеко от допустимого (относительно конкретного ограничения). Если значение вспомогательной переменной больше нуля, то данное решение не выполняет определённое ограничение, а значит не является допустимым.

То есть ненулевое значение дополнительной переменной может (но не должно) сигнализировать о неоптимальности решения. Ненулевое значение вспомогательной переменной сигнализирует о недопустимости решения.

[править] Фазы решения

После того, как было модифицировано условие, создаётся вспомогательная целевая функция. Если вспомогательные переменные были обозначены, как yi, i∈{1, .., k}, то вспомогательную функцию определим, как

.

После этого проводится обыкновенный симплекс-метод относительно вспомогательной целевой функции. Поскольку все вспомогательные переменные увеличивают значение z', в ходе алгоритма они будут поочерёдно выводится из базиса, при этом после каждого перехода новое решение будет всё ближе к множеству допустимых решений.

Когда будет найдено оптимальное значение вспомогательной целевой функции, могут возникнуть две ситуации:

оптимальное значение z' больше нуля. Это значит, что как минимум одна из вспомогательных переменных осталась в базисе. В таком случае можно сделать вывод, что допустимых решений данной задачи линейного программирования не существует.

оптимальное значение z' равно нулю. Это означает, что все вспомогательные переменные были выведены из базиса, и текущее решение является допустимым.

Во втором случае мы имеем допустимый базис, или, иначе говоря, исходное допустимое решение. Можно проводить дальнейшую оптимизацию с учётом исходной целевой функции, при этом уже не обращая внимания на вспомогательные переменные. Это и явлется второй фазой решения.

[править] Модифицированный симплекс-метод

[править] Двойственный симплекс-метод

[править] Литература

Хемди А. Таха Глава 3. Симплекс-метод // Введение в исследование операций = Operations Research: An Introduction. — 7-е изд. — М.: «Вильямс», 2007. — С. 95-141. — ISBN 0-13-032374-8

Акулич И.Л. Глава 1. Задачи линейного программирования // Математическое программирование в примерах и задачах. — М.: Высшая школа, 1986. — 319 с. — ISBN 5-06-002663-9.

п··р

Методы оптимизации

Одномерные Метод золотого сечения • Метод деления пополам • Метод дихотомии • Метод парабол • Метод равномерного поиска (перебора) • Метод равномерного блочного поиска • Метод троичного поиска

Прямые методы Метод Гаусса • Метод Нелдера — Мида • Метод конфигураций • Метод Розенброка • Метод сопряжённых направлений • Метод Хука — Дживса

Первого порядка Градиентный спуск • Метод покоординатного спуска • Метод сопряжённых градиентов

Второго порядка Метод Ньютона • Метод Ньютона-Рафсона

Стохастические Дифференциальная эволюция • Имитация отжига

Методы линейного

программирования Метод эллипсоидов • Симплекс-метод • Метод потенциалов

Это незавершённая статья по математике. Вы можете помочь проекту, исправив и дополнив её.