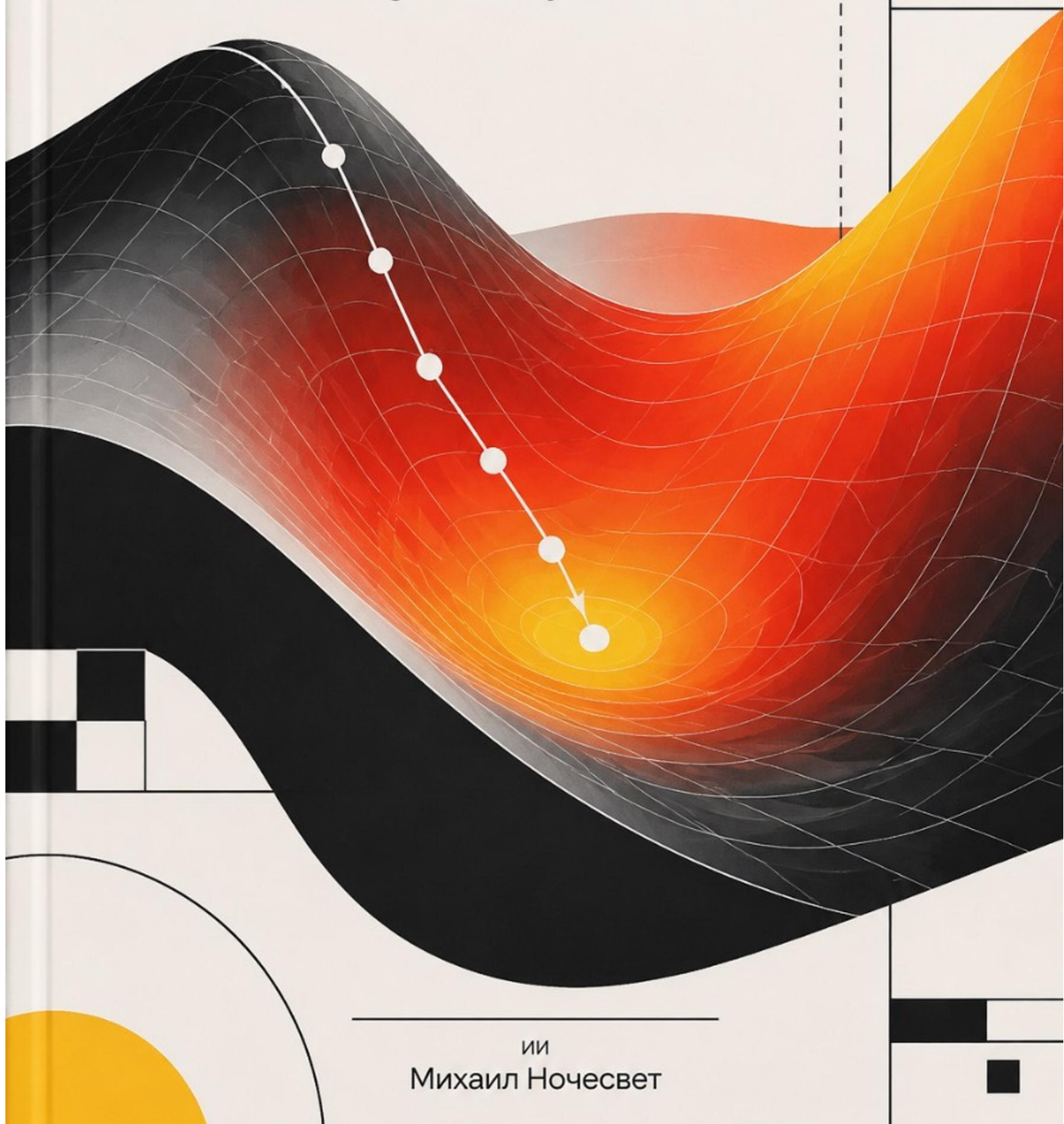


Вектор ОПТИМУМА:

Градиентный метод
для поиска лучших решений



и
Михаил Ночесвет

Михаил Ночесвет

**Вектор оптимума: Градиентный
метод для поиска лучших решений**

«Автор»

2026

Ночесвет М.

Вектор оптимума: Градиентный метод для поиска лучших решений
/ М. Ночесвет — «Автор», 2026

Утро в пекарне: очередь растёт, хлеб закончился, вчерашние булочки остались. План снова строили на памяти и «кажется». Как заменить догадки проверяемым решением? Градиентный метод превращает поиск лучшего варианта в цепочку шагов: задать цель и ограничения, увидеть рельеф задачи и двигаться к меньшей ошибке. Слишком резкий шаг уводит в сторону, осторожный тормозит, а малый градиент ещё не доказывает, что найден глобальный минимум. Вы разберёте градиентный спуск, выбор шага и остановки, масштабирование, мини-пакеты, импульс и адаптивный шаг, поиск с ограничениями. Научитесь проверять градиенты и код, оценивать результат в задачах машинного обучения и инженерии. Для студентов, аналитиков и разработчиков, которым нужны не догадки, а решения с понятной логикой и границами применимости. Книга создана с помощью ИИ. Обложка: GPT Image - Лицензия

© Ночесвет М., 2026

© Автор, 2026

Содержание

От практической задачи к задаче оптимизации: переменные, цель и ограничения	6
Рельеф целевой функции: уровни, выпуклость и обусловленность	12
Градиент как компас: производные, направления и цепное правило	17
Градиентный спуск: от первой итерации к работающему алгоритму	22
Конец ознакомительного фрагмента.	25

Михаил Ночесвет
Вектор оптимума: Градиентный
метод для поиска лучших решений

Вектор оптимума: Градиентный метод для поиска лучших решений
Михаил Ночесвет

От практической задачи к задаче оптимизации: переменные, цель и ограничения

В шесть утра в пекарне пахнет дрожжами, кофе и лёгкой паникой. У дверей уже ждут первые покупатели. На полке, где должен лежать хлеб, — пустота. На соседнем подносе остались вчерашние булочки: к обеду они, скорее всего, станут ещё менее желанными. Хозяйка пекарни Марина смотрит на очередь и пытается вспомнить, сколько всего испекли в прошлый вторник. «Кажется, хлеба было мало. А круассанов, наоборот, многовато».

Вчера она планировала выпечку примерно так же. И позавчера. Каждый раз — на основе опыта, памяти и того самого «кажется». Опыт полезен, но память сглаживает детали: очередь в один день запоминается лучше, чем тихое утро в другой. Марине нужно не просто «работать лучше». Ей нужно каждое утро решить, сколько изготовить хлеба, булочек и круассанов, чтобы заработать, не остаться без популярных товаров и не выбросить остатки.

Это уже задача, которую можно разобрать. Но начинать следует не с градиента, формулы или программы. Начинать нужно с вопроса: **какое именно решение мы выбираем?**

1. Сначала — решение, потом математика

Марина управляет количеством выпечки на сегодня. Обозначим число произведённых буханок через (qh) , булочек — через (qb) , круассанов — через (qc) . Буква (q) напоминает о количестве, а индексы помогают не путать продукты.

Но произведённое не всегда равно проданному. Если испечь 30 булочек, а купить готовы только 24, шесть останутся. Поэтому введём ещё три переменные: (sh) , (sb) , (sc) — сколько единиц каждого товара будет продано.

Так мы отделяем два разных решения и события: **сколько изготовить** и **сколько продать**. Это не педантизм ради букв. Именно в промежутке между ними возникают непроданные остатки.

Здесь важно отличать переменные решения от параметров модели. Переменные — то, что Марина выбирает. Параметры — данные, которые она пока считает известными: цена продукта, расход муки, доступное время. Например, цену хлеба Марина сегодня не выбирает в задаче о плане выпечки; она получает её из действующего прайс-листа. Поэтому цена — параметр, а количество буханок — переменная.

2. Что известно, а что нужно решить

Для первого приближения Марина собирает данные за утреннюю смену. Расход времени указан как среднее число минут производственной работы на одну единицу продукции. В реальной пекарне часть операций идёт параллельно; пока мы объединим их в один показатель, чтобы построить понятную начальную модель.

Продукт

Цена продажи, руб.

Переменные затраты, руб.

Время, мин.

Мука, кг

Масло, кг

Прогноз спроса, шт.

Хлеб

160

60

12
0,50
0,02
40

Булочка
70
25
4
0,10
0,02
80

Круассан
130
85
8
0,12
0,06
35

На смену у Марины есть не более 360 минут производственного времени, 24 килограммов муки и 3 килограммов масла. Это **параметры ресурсов**: в плане на сегодня они заданы. Если завтра поставка масла задержится, число изменится — и вместе с ним изменится задача.

Ещё один параметр — прогноз спроса. Он говорит, сколько товара, по оценке Марины, удастся продать. Это не точное пророчество и не обязательно жёсткий закон природы; на первом шаге мы просто не планируем продаж больше, чем ожидаемый спрос. Позже можно будет учесть неопределённость подробнее.

3. Цель должна измеряться

«Хочу, чтобы день прошёл лучше» — хорошее человеческое желание, но плохая математическая цель. Модель должна понимать, что именно считать успехом. Для Марины естественная первая цель — максимизировать прибыль от сегодняшней выпечки.

Если единица товара продана, пекарня получает выручку. Если она произведена, возникают затраты — даже когда продукт так и не купили. Поэтому выручка зависит от продаж (s) , а производственные расходы — от количества изготовленного (q) :

$$P = 160sh + 70sb + 130sc - 60qh - 25qb - 85qc.$$

Здесь P — прибыль за смену, измеряемая в рублях. Слово «максимизировать» означает: среди допустимых планов выбрать тот, у которого значение P наибольшее.

Почему нельзя просто максимизировать выручку? Потому что высокая цена ещё не означает высокую прибыль. Один круассан приносит 130 рублей выручки, а одна булочка — 70. Но переменные затраты на круассан составляют 85 рублей, на булочку — 25. После вычета затрат в обоих случаях остаётся по 45 рублей. При этом круассан требует вдвое больше времени и втрое больше масла. Если поставить целью одну только выручку, модель будет считать круассан гораздо привлекательнее, чем он выглядит с точки зрения заработка и ресурсов.

Это типичная ловушка: оптимизировать измеримый показатель, который лишь похож на настоящее желание. Иногда компании максимизируют число заказов, хотя важна прибыль; люди сокращают время поездки, не замечая стоимости дороги; пекарня печёт больше, полагая, что «больше продаж» автоматически означает «лучший день».

У цели должен быть ясный смысл. Если Марина хочет прежде всего увеличить прибыль — мы записываем прибыль. Если для неё критично, чтобы к обеду на полке был хлеб, это тоже важно, но такую заботу нельзя незаметно растворить в слове «успех». Её нужно выразить отдельно.

4. Ограничения очерчивают допустимое

Целевая функция отвечает на вопрос: **какой результат предпочитаем?** Ограничения отвечают на другой вопрос: **какие планы вообще возможны?**

Например, на производство нельзя потратить больше имеющегося времени:

$$\lfloor 12qh+4qb+8qc \leq 360. \rfloor$$

Каждый коэффициент показывает, сколько минут требуется на одну единицу продукта. Если Марина планирует 10 буханок, это 120 минут; 20 булочек — ещё 80; 5 круассанов — ещё 40. Левая часть суммирует время всех изделий. Знак $\leq$ означает: можно потратить меньше доступного, но нельзя больше.

По той же логике записываются запасы ингредиентов:

$$\lfloor 0\{, \}50qh+0\{, \}10qb+0\{, \}12qc \leq 24 \rfloor$$

для муки и

$$\lfloor 0\{, \}02qh+0\{, \}02qb+0\{, \}06qc \leq 3 \rfloor$$

для масла.

Продать больше, чем изготовлено, нельзя:

$$\lfloor sh \leq qh, \quad sb \leq qb, \quad sc \leq qc. \rfloor$$

А прогноз спроса ограничивает возможные продажи:

$$\lfloor sh \leq 40, \quad sb \leq 80, \quad sc \leq 35. \rfloor$$

Наконец, количества не могут быть отрицательными. И в обычной пекарне нельзя выпустить 3,7 буханки, поэтому для точной модели количества целые:

$$\lfloor qh, qb, qc, sh, sb, sc \in \mathbb{Z} \rfloor.$$

Все эти условия вместе и есть ограничения модели.

Обратите внимание: ограничение не обязательно выражает нехватку. Оно может отражать обязательство. Если Марина решила, что хлеб должен оставаться в ассортименте и каждый день нужно произвести не менее 20 буханок, можно добавить условие $(qh \geq 20)$. Но это уже осознанный выбор: такой минимум защищает ассортимент, зато может увеличить остатки. Математика не решает, важен ли хлеб покупателям. Она помогает точно показать, что произойдёт, если Марина внесёт это требование в план.

5. Собираем модель целиком

Полная начальная постановка выглядит так:

$$\begin{aligned} \max \quad & 160sh+70sb+130sc-60qh-25qb-85qc, \\ \text{при условиях} \quad & 12qh+4qb+8qc \leq 360, \\ & 0\{, \}50qh+0\{, \}10qb+0\{, \}12qc \leq 24, \\ & 0\{, \}02qh+0\{, \}02qb+0\{, \}06qc \leq 3, \\ & 0 \leq sh \leq qh, \quad 0 \leq sb \leq qb, \quad 0 \leq sc \leq qc, \\ & sh \leq 40, \quad sb \leq 80, \quad sc \leq 35, \\ & qh, qb, qc, sh, sb, sc \in \mathbb{Z}. \end{aligned}$$

У этой записи есть простое прочтение: выбери целые объёмы производства и продаж так, чтобы не превысить время и запасы, не продавать несуществующий товар и не рассчитывать на спрос выше прогноза; среди всех таких планов найди тот, который даёт наибольшую прибыль.

Модель ещё не говорит, что Марина обязана довериться прогнозу. Она показывает, **какой вопрос мы задаём**, опираясь на эти данные и допущения. Это различие принципиально: оптимизатор не превращает сомнительные исходные числа в надёжные. Точность решения не может быть выше качества сведений, на которых оно построено.

6. Карточка модели: проверка перед расчётом

Прежде чем искать лучший план, Марина сводит постановку в короткую карточку.

Поле карточки

Модель пекарни

Что выбираем

Сколько произвести и сколько продать каждого продукта

Что улучшаем

Прибыль за утреннюю смену, рубли

Что нельзя нарушать

Запасы времени, муки и масла; продажи не выше производства и прогноза спроса

Единицы измерения

Штуки, минуты, килограммы, рубли

Основные допущения

Цены и затраты постоянны в течение смены; время на единицу усреднено; спрос ограничен прогнозом; остатки не имеют ценности

Какие данные неизвестны или неточны

Фактический спрос, изменение времени работы, возможные скидки, стоимость утилизации остатков

Последняя строка особенно важна. Если круассаны иногда распродают после скидки, их остатки не обязательно имеют нулевую ценность. Если утилизация стоит денег, в целевой функции нужен дополнительный расход. Если продажи зависят от погоды, дня недели или соседнего кафе, один прогноз может оказаться слишком грубым.

У прогноза спроса есть и менее заметная ловушка. Если хлеб закончился к десяти утра, данные о продажах покажут, сколько хлеба **успели продать**, а не сколько покупатели хотели бы купить за весь день. Пустая полка скрывает неудовлетворённый спрос. Поэтому Марине полезно отдельно записывать случаи, когда товар закончился до конца смены. Иначе модель может сделать аккуратный вывод из неполных данных: «сорока буханок не нужно», потому что в журнале продаж их никогда и не было.

7. Проверка здравого смысла

До сложных расчётов стоит проверить, не противоречит ли модель сама себе. Например, план из 80 булочек и 3 буханок требует:

- 356 минут: $(80 \cdot 4 + 3 \cdot 12)$;
- 9,5 килограмма муки: $(80 \cdot 0{,}1 + 3 \cdot 0{,}5)$;
- 1,66 килограмма масла: $(80 \cdot 0{,}02 + 3 \cdot 0{,}05)$.

Он укладывается в заданные запасы и не превышает прогноз спроса. Если считать, что всё произведённое продаётся, прибыль такого плана составит $(80 \cdot 45 + 3 \cdot 100 = 3900)$ рублей. Это не доказательство, что план лучший. Это лишь проверка, что числа и ограничения прочитаны правильно.

Полезны и другие короткие тесты.

1. Проверить размерности. В уравнении времени число минут на штуку умножается на количество штук, а результат измеряется минутами. Если в одной части ограничения внезапно сложились рубли и килограммы, ошибка видна ещё до запуска алгоритма.

2. Проверить знаки. Дополнительные продажи обычно увеличивают прибыль, а дополнительные производственные расходы — уменьшают. Если затраты в целевой функции стоят со знаком «плюс», стоит остановиться и выяснить почему.

3. Проверить крайний случай. Нулевое производство должно быть допустимым, если в модели нет обязательного минимума. Если оно запрещено, должна быть понятная причина.

4. Сравнить модель с реальностью. Может ли пекарь действительно изготовить рассчитанное количество за указанное время? Есть ли в таблице упаковка, уборка, прогрев печи? Не учтено ли одно и то же время дважды?

Так модель проходит не лабораторный, а человеческий контроль: хозяин должен узнавать в ней свою работу.

8. Где заканчиваются данные и начинаются решения

Числа в таблице не одинаково надёжны. Цена продажи известна точно. Средний расход муки, вероятно, довольно стабилен. А спрос на завтра — оценка. Полезно не складывать их в одну кучу, будто все они — одинаково твёрдые факты.

Если спрос на круассаны колеблется от 15 до 35 штук, можно использовать осторожный прогноз, сравнить несколько сценариев или задать запас на случай высокого спроса. Если время на изготовление зависит от загрузки печи, одного ограничения по минутам может быть недостаточно: понадобится отдельная мощность оборудования. Если товар быстро теряет привлекательность, для разных изделий могут потребоваться разные ограничения на запас или срок продажи.

Это не означает, что первая модель обязана описать каждую крошку и минуту. Слишком подробная модель бывает так же неудобна, как карта, на которой нарисован каждый камешек. Хорошее начало — достаточная точность для важного решения и явное перечисление того, что пока упрощено.

Есть ещё одно существенное упрощение: количества в пекарне целые, а базовый градиентный метод работает с непрерывными переменными и предполагает, что функцию можно гладко менять. Нельзя бездумно выдать результат «21,6 буханки», округлить его до 22 и считать задачу решённой: после округления можно нарушить ограничение по времени или ингредиентам. Для целочисленных решений нужны специальные приёмы, допустимое приближение или метод, подходящий для целых переменных. Но выбирать способ решения имеет смысл только после того, как сама задача сформулирована честно.

9. Переносим способ на домашнюю задачу

Тот же подход годится не только для пекарни. Представьте, что вы планируете меню на рабочую неделю. Не начинайте с вопроса «как питаться идеально?». Сформулируйте решение конкретнее.

1. Что выбираем? Количество порций каждого блюда.

2. Что измеряем? Общую стоимость покупок или время готовки — это возможные цели.

3. Что нельзя нарушать? Бюджет, доступное время, необходимые ограничения рациона.

4. Какие данные известны не точно? Цена в магазине, срок хранения, вероятность того, что в пятницу вы действительно захотите приготовить запланированное.

Если цель — минимизировать расходы, это не то же самое, что ограничить бюджет. Первая формулировка говорит, что желательно улучшить; вторая — что нельзя превысить. И если вы одновременно хотите дёшево, быстро, разнообразно и без остатков, придётся решить, что важнее, или выразить часть желаний в виде обязательных условий.

Попробуйте прямо сейчас выбрать небольшую бытовую задачу — например, собрать план поездки, распределить время подготовки к экзамену или решить, сколько продуктов купить. Запишите одной строкой: что вы выбираете, что считаете успехом, какие условия нельзя нарушить и какие данные лишь предполагаете. Если для ответа приходится использовать слова «удобно», «много» или «достаточно», спросите себя: как я узнаю, что именно это произошло?

От догадки — к ясному вопросу

Марина не получила волшебной цифры, которая отменяет опыт. Она получила нечто полезнее: способ проверить опыт числами. Теперь видно, где находится выбор, какие сведения влияют на результат и какие требования нельзя переступить. Можно спорить не о том, «мало ли круассанов», а о конкретных вещах: насколько надёжен прогноз спроса, сколько масла останется и стоит ли вводить минимум хлеба ради покупателей.

Так практическая задача становится задачей оптимизации: мы называем переменные решения, выбираем измеримую цель, отделяем параметры от переменных и записываем ограничения с понятными единицами. Это ещё не поиск ответа. Это точное описание вопроса, на который ответ предстоит искать.

А когда вопрос сформулирован, появляется следующий: как устроен ландшафт этой цели — какие значения лучше, где лежат допустимые решения и насколько легко двигаться к улучшению? Именно с этого начинается разговор о рельефе целевой функции.

Рельеф целевой функции: уровни, выпуклость и обусловленность

В 23:40 команда инженеров в третий раз запускает настройку модели. На экране — две ручки. Первая почти не меняет ошибку: её можно крутить долго, и показатель плавно ползёт вниз. Вторая будто связана с пружиной: чуть сдвинешь — ошибка резко меняется, а иногда и подсказывает. «Значит, второй параметр важнее», — говорит кто-то. Но инженер, который смотрит на график, качает головой: возможно, параметр не важнее. Возможно, поверхность задачи просто устроена так, что в одном направлении она пологая, а в другом — крутая.

Представьте, что целевая функция — это рельеф, а каждая настройка модели задаёт координату на карте. Высота в точке — значение ошибки. Наша задача — найти низину. Но карта не видна напрямую: мы можем лишь менять параметры и измерять результат. Линии уровня помогают восстановить её форму. Именно форма рельефа объясняет, почему градиентный метод в одной задаче скользит к минимуму уверенно, а в другой мечется, словно пытается пройти по узкой тропе в тумане.

Контурные на невидимой карте

Линия уровня соединяет все точки, где целевая функция имеет одно и то же значение:

$$\{ (x,y) : f(x,y)=c \}$$

На географической карте линии уровня соединяют точки с одинаковой высотой. На карте оптимизации — точки с одинаковой ошибкой, затратами или риском. Если рядом с текущей точкой линии расположены тесно, небольшое перемещение в направлении поперёк них заметно меняет значение функции. Если линии разрежены, рельеф в этом направлении пологий.

Для простейшей круглой чаши

$$f(x,y)=\frac{1}{2}(x^2+y^2)$$

линии уровня — окружности:

$$x^2+y^2=2c.$$

Чем ближе к центру, тем ниже значение функции. Со всех сторон к центру ведёт одинаковый склон. Это симметричная, удобная для поиска местность: ни одному направлению не отдано предпочтение.

Для настоящей инженерной задачи карта обычно сложнее. Один параметр может изменяться тысячами, другой — десятками долями; параметры могут влиять на ошибку совместно; изменение одного способно частично компенсировать изменение другого. Поэтому один и тот же уровень ошибки иногда образует не круг, а вытянутый овал. Такой контур говорит: вдоль длинной оси можно пройти далеко, почти не меняя значение функции, а поперёк короткой оси — совсем немного.

Есть важная оговорка: линия уровня — не сама долина, а лишь её горизонтальный контур. Ещё полезнее смотреть на область под уровнем,

$$\{ (x,y) : f(x,y) \leq c \},$$

то есть на все точки, где значение не выше c . Для выпуклой функции такие области выпуклы. А вот отдельная линия $\{f=c\}$ не обязана быть выпуклой или даже выглядеть как одна простая замкнутая петля.

В точке, где функция гладкая и градиент не равен нулю, градиент перпендикулярен линии уровня. Если идти вдоль контура, значение не меняется; градиент указывает направление, в котором оно растёт быстрее всего. Позже мы превратим это наблюдение в подробный разговор о производных и направлениях. Пока достаточно удерживать геометрическую мысль:

градиент смотрит поперёк контура, а не обязательно прямо к ближайшей на глаз точке минимума.

Что даёт выпуклость

Слово «выпуклость» описывает не просто красивую чашу на картинке, а точное свойство функции. Функция выпукла на некоторой области, если для любых двух точек a и b и любого числа λ выполняется условие

$$f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda)f(b).$$

Если соединить значения функции в точках a и b прямой, график выпуклой функции между ними не поднимется выше этой прямой. В одном измерении это похоже на чашу. В нескольких измерениях такой проверки нужно придерживаться вдоль каждого возможного отрезка внутри рассматриваемой области.

Зачем это оптимизатору? У выпуклой функции нет ложных локальных минимумов: всякий локальный минимум является глобальным. Иными словами, если мы нашли точку, из которой ни в одном достаточно близком направлении нельзя спуститься ниже, то ниже неё нигде на выпуклой области уже не будет. Это очень сильная гарантия.

Но выпуклость не означает, что минимум обязательно один. Например, функция $f(x,y)=x^2$ выпукла, но её минимум достигается во всех точках прямой $x=0$. Для единственного минимума нужны дополнительные условия. Строгая выпуклость гарантирует, что минимум, если он существует, не может быть более чем в одной точке. Сильная выпуклость добавляет ещё более полезное свойство: у функции есть минимальная кривизна, то есть она не становится совершенно плоской ни в одном направлении.

Здесь легко смешать два разных понятия. Гладкость говорит о том, насколько регулярно меняется функция и её производная. Выпуклость — о форме функции и отсутствии ложных впадин. Гладкая функция не обязательно выпукла: у неё могут быть несколько ям, гребни и седловые точки. Выпуклая функция, в свою очередь, может иметь излом и не быть гладкой. Для работы градиентного метода важны оба свойства, но одно не следует из другого.

Гарантии выпуклости относятся к области, которую мы рассматриваем. Если задача ограничена, значение имеют не только функция, но и множество допустимых решений: для стандартных глобальных гарантий оно тоже должно иметь подходящую выпуклую структуру. А если функция невыпукла, градиентный метод всё ещё может быть полезен, однако обещать, что найденная впадина окажется самой глубокой во всей местности, уже нельзя. Он может остановиться в локальном минимуме; также его способны задержать плоский участок или седловая область.

Круглая чаша и узкая долина

Вернёмся к команде инженеров. Чтобы увидеть разницу между удобной и неудобной геометрией, сравним круглую чашу с вытянутой долиной. В круглой чаше линии уровня похожи на окружности, а склон во всех направлениях меняется сходным образом. В долине контуры растянуты: в одном направлении функция меняется резко, в другом — очень медленно.

Покажем это на двумерном квадратичном примере, который можно проследить вручную. Введём две взаимно перпендикулярные координаты:

$$u = \frac{x+y}{\sqrt{2}}, \quad v = \frac{x-y}{\sqrt{2}},$$

и зададим функцию

$$f(x,y) = \frac{1}{2}(u^2 + 100v^2).$$

Координата u идёт вдоль долины, а v — поперёк неё. Вдоль долины кривизна равна 1 , поперёк — 100 . Поэтому для одного и того же уровня линии образуют эллипсы: длинная ось направлена вдоль u , короткая — вдоль v . Отношение их полуосей равно $10:1$. Долина проходит по диагонали $x=y$ и сужается к минимуму в начале координат.

Теперь запустим градиентный спуск. В координатах (u,v) его шаги особенно наглядны:

$$u_{k+1} = (1-\alpha)u_k, \quad v_{k+1} = (1-100\alpha)v_k,$$

где α — размер шага. Начнём из точки $(x, y) = (1, 0)$. Тогда $u_0 = v_0 = 1/\sqrt{2}$. Возьмём $\alpha = 0{,}018$. Множитель вдоль долины будет $(0{,}982)$: продвижение медленное. Поперёк долины множитель равен $(-0{,}8)$: знак меняется, и точка перескакивает через линию $x = y$.

После первого шага координаты примерно равны $(0{,}09; 0{,}89)$, после второго — $(0{,}80; 0{,}16)$. Точка ходит то на одну, то на другую сторону узкой долины. При этом значение функции уменьшается — алгоритм не заблудился, но движение выглядит зигзагообразным. Поперечное колебание постепенно затухает, а продвижение к минимуму вдоль долины остаётся неторопливым.

Почему не взять гораздо меньший шаг, чтобы не перескакивать через стенки долины? Можно. Но тогда каждый шаг будет безопаснее поперёк, а продвижение вдоль длинного направления станет ещё медленнее. Если, напротив, выбрать шаг, подходящий для пологого направления, можно сделать скачок слишком большим в направлении крутой стенки. При чрезмерном шаге алгоритм способен не просто перескакивать через долину, а расходиться.

В этом и состоит практическая неприятность вытянутых контуров: одному размеру шага приходится обслуживать направления с очень разной кривизной. Круглая чаша прощает больше. Узкая долина требует осторожности и часто выигрывает от изменения масштаба переменных, подбора шага или более совершенной схемы поиска.

Обусловленность: насколько сильно вытянута задача

Для квадратичной функции кривизну описывает матрица вторых производных — матрица Гессе. У нашей долины её собственные значения равны (1) и (100) : это кривизны вдоль главных направлений. Их отношение

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$$

называют числом обусловленности. В данном примере $\kappa = 100$. У круглой чаши обе кривизны одинаковы, поэтому $\kappa = 1$: задача хорошо обусловлена. Чем сильнее вытянуты эллипсы, тем больше число обусловленности и тем обычно труднее базовому градиентному спуску быстро приблизиться к минимуму.

Для гладкой сильно выпуклой функции часто рассматривают границы кривизны: минимальную $(\mu > 0)$ и максимальную (L) . Их отношение (L/μ) играет роль глобального числа обусловленности. При стандартных предположениях скорость простого градиентного спуска по числу итераций ухудшается примерно пропорционально этому отношению. Это не означает, что каждую задачу с большим числом обусловленности невозможно решить. Это означает, что геометрия может заставить простой алгоритм делать много осторожных шагов.

Если минимальная кривизна равна нулю, ситуация становится особенной. Например, у $f(x, y) = x^2$ функция совсем не меняется вдоль (y) . В этом направлении нет ни склона, ни единственного места, куда нужно прийти. Для такого случая конечное число обусловленности в приведённом смысле не получается: минимальная кривизна отсутствует. Это не обязательно ошибка задачи, но важный сигнал — решение может быть неединственным, а движение в плоском направлении не будет уточнять ответ.

Число обусловленности зависит и от того, как заданы координаты. Если один параметр записан в метрах, а другой — в миллиметрах, геометрия на графике может выглядеть искусственно растянутой. Даже одинаковая физическая переменная в разных единицах получает разные численные размеры. Поэтому перед выводом о «важности» параметра стоит проверить масштабирование. Нормирование признаков, перевод параметров в сопоставимые единицы и методы преобуславливания способны заметно улучшить маршрут поиска. Но они не превращают автоматически невыпуклую задачу в выпуклую и не меняют смысл цели, если преобразование выполнено корректно.

Как читать карту на практике

Визуальная диагностика полезна, если рисовать её аккуратно. Сначала выберите два параметра, зафиксируйте остальные и измерьте значение функции на сетке вокруг интересующей точки. Затем постройте линии одинакового значения. Для сравнения разных карт используйте сопоставимые интервалы уровней и проверьте масштаб осей: круг, нарисованный при неодинаковом масштабе по горизонтали и вертикали, легко принять за эллипс.

Ищите три признака. Первый — вытянутость контуров: она указывает на направления с различной чувствительностью. Второй — сгущение линий: оно говорит о более резком изменении функции поперёк них. Третий — число и расположение отдельных впадин. Несколько замкнутых семейств контуров могут означать несколько областей притяжения и повышенную зависимость результата от начальной точки.

Однако двумерная картинка — только срез многомерной задачи. Пока меняются два параметра, остальные обычно удерживаются фиксированными. В другом срезе рельеф может выглядеть иначе; важное направление может быть комбинацией сразу нескольких параметров. Поэтому контурный график — не полная карта, а полезный снимок участка местности.

Наконец, величина градиента сама по себе не доказывает, что параметр «важнее». Производная зависит от единиц измерения: если заменить доллары на центы, численное значение производной изменится. Для понимания чувствительности важны и масштаб изменения параметра, и форма уровней, и то, в какой области мы находимся.

Упражнение: оцените маршрут

Представьте, что вы видите три карты, на каждой нанесены линии равного значения целевой функции. Не пытаясь ещё запускать оптимизатор, предскажите, где простой градиентный спуск будет работать легче, а где потребуются дополнительные меры.

Карта

Что видно на линиях уровня

Ожидаемая трудность

А

Вокруг одного центра расположены почти круглые вложенные контуры
Относительно лёгкий поиск: направления меняются сходным образом

Б

Контуры — длинные узкие эллипсы с отношением осей около $\sqrt{30:1}$

Высокая обусловленность: вероятны медленный прогресс вдоль долины и колебания поперёк

В

Видно несколько отдельных областей замкнутых контуров, а между ними — седловидные проходы

Функция, вероятно, невыпукла: результат может зависеть от начальной точки, глобальный минимум не гарантирован

Проверьте себя: карта А не обещает мгновенного решения, но её геометрия наиболее благоприятна. Карта Б прежде всего предупреждает о трудности масштаба и шага. Карта В заставляет задать другой вопрос: в какую именно впадину попадёт алгоритм и существует ли более глубокая? Контуры помогают сформулировать прогноз, но сами по себе не доказывают, что функция выпукла или что найденная точка глобально лучшая.

Рельеф целевой функции — не украшение к формулам, а часть самой задачи. Линии уровня показывают, где ошибка одинакова; выпуклость объясняет, когда локальный успех

означает глобальный; обусловленность раскрывает, почему узкие долины замедляют поиск. Когда один параметр будто движется свободно, а другой требует усилий, полезно смотреть не только на их значения, но и на карту, которую они вместе вычерчивают. А чтобы научиться понимать, почему градиент выбирает именно такое направление шага, теперь пора рассмотреть его как компас, построенный из производных.

Градиент как компас: производные, направления и цепное правило

Аналитик меняет один параметр модели наугад: добавляет к нему сотую долю, запускает расчёт и ждёт. Ошибка уменьшается. Хороший знак — попробуем прибавить ещё. Теперь ошибка растёт. Он возвращает параметр назад, пробует другой шаг, затем переключается на соседний параметр. День уходит на серию опытов, каждый из которых сообщает лишь одно: «здесь стало лучше» или «здесь стало хуже». Но насколько лучше? И куда двигаться дальше?

Есть способ заменить перебор измерением. Не просто слегка менять параметр, а выяснять, как быстро меняется целевая функция при таком изменении. Это и есть первый шаг от догадок к градиенту.

Сначала — скорость изменения

Представим, что ошибка модели зависит только от одного параметра w , а её значение описывается простой функцией:

$$L(w) = (w-3)^2.$$

Если сейчас $w=1$, ошибка равна 4. Аналитик увеличивает параметр на 0.1: новое значение ошибки — 3.61. Изменение параметра составило 0.1, а изменение ошибки — -0.39. Средняя скорость изменения на этом маленьком отрезке равна

$$\frac{3.61-4}{0.1} = -3.9.$$

Знак минус важен: при увеличении w ошибка уменьшается. Если сделать шаг ещё меньше, отношение изменения ошибки к изменению параметра будет приближаться к -4. Это и есть производная $L'(1)$.

Производная отвечает на вопрос: **как быстро и в какую сторону меняется функция, если чуть-чуть изменить аргумент?** Она не говорит, чему равна сама функция. В нашем примере значение функции в точке — 4, а производная — примерно -4. Первое описывает высоту ошибки, второе — наклон поверхности в выбранной точке.

Формально производную можно представить как предел отношений малых изменений:

$$L'(w) = \lim_{h \rightarrow 0} \frac{L(w+h) - L(w)}{h}.$$

В практической работе мы не обязаны вычислять этот предел каждый раз с нуля. Часто производную удастся получить аналитически. Но её смысл остаётся тем же: это локальная скорость изменения. Если производная положительна, увеличение параметра вблизи данной точки повышает функцию. Если отрицательна — снижает. Если близка к нулю, функция в этом направлении меняется медленно, хотя это ещё не доказывает, что найден минимум.

От одного параметра к карте чувствительности

Обычно у модели не один параметр, а целый набор: веса, смещения, коэффициенты, настройки. Обозначим их вектором $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_n)$. Целевая функция теперь зависит сразу от нескольких переменных: $L(\boldsymbol{\theta})$.

Чтобы понять её поведение, можно начать по очереди менять каждый параметр, оставляя остальные неизменными. Так мы получим частные производные:

$$\frac{\partial L}{\partial \theta_1}, \quad \frac{\partial L}{\partial \theta_2}, \quad \dots, \quad \frac{\partial L}{\partial \theta_n}.$$

Каждая частная производная — отдельный показатель чувствительности. Например, $\frac{\partial L}{\partial \theta_2}$ сообщает, как изменится ошибка, если немного изменить θ_2 , не трогая остальные параметры. Это похоже на то, как инженер проверяет отдельные регуляторы на пульте: один поворачивает, все остальные пока оставляет на месте.

Соберём эти числа в вектор — **градиент**:

$$\nabla L(\boldsymbol{\theta}) = \left(\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \dots, \frac{\partial L}{\partial \theta_n} \right).$$

Важное различие: значение функции — число, частная производная — число, градиент — вектор из частных производных. Если целевая функция показывает высоту местности, то градиент в каждой точке описывает её локальный наклон сразу по всем координатам.

Для небольшого изменения параметров $\Delta \boldsymbol{\theta}$ изменение функции можно приблизительно оценить так:

$$\Delta L \approx \nabla L \cdot \Delta \boldsymbol{\theta}.$$

Точка между векторами обозначает скалярное произведение. Эта формула — компактный перевод с языка геометрии на язык чисел: изменение ошибки зависит от того, насколько направление шага совпадает с градиентом.

Стрелка на ландшафте

Вообразим рельефную карту, где линии соединяют точки с одинаковой высотой. Для целевой функции такими линиями уровня будут все наборы параметров, при которых ошибка одинакова. Если идти вдоль одной такой линии, значение функции не меняется.

Теперь представим, что мы немного сдвинулись вдоль линии уровня. Изменение функции равно нулю, а значит, градиент не может иметь составляющую вдоль этого движения. Иначе движение в направлении градиента меняло бы высоту. Поэтому градиент пересекает линию уровня под прямым углом: он направлен по нормали к ней.

Но почему именно в сторону самого быстрого роста? Возьмём любое направление длины один — то есть направление, в котором мы сравниваем одинаково короткие шаги. Скорость изменения функции в нём равна скалярному произведению градиента на этот единичный вектор. По неравенству Коши — Буняковского это произведение максимально, когда направление шага совпадает с направлением градиента. Иными словами, градиент указывает туда, где функция растёт быстрее всего. Противоположное направление даёт самое быстрое локальное убывание.

Вот почему, если мы хотим уменьшить ошибку, двигаться нужно **против градиента**. Минус перед градиентом не случайная договорённость и не магический рецепт. Это геометрическое следствие того, как устроена скорость изменения функции.

Рассмотрим простой ландшафт:

$$f(x, y) = x^2 + 4y^2.$$

Его линии уровня — эллипсы: на каждой из них значение f одинаково. Градиент равен $\nabla f(x, y) = (2x, 8y)$.

В точке $(1, 1)$ он составляет $(2, 8)$. Стрелка градиента направлена перпендикулярно эллипсу через эту точку и показывает самый крутой подъём. Для спуска мы выбираем противоположное направление: $(-2, -8)$.

Можно сделать небольшой шаг длиной, заданной числом $\alpha = 0.05$, вдоль отрицательного градиента:

$$(1, 1) - 0.05(2, 8) = (0.9, 0.6).$$

В исходной точке функция равна 5 . В новой точке она равна $0.9^2 + 4 \cdot 0.6^2 = 2.25$. Значение уменьшилось. Градиент здесь сработал как компас: он сообщил, куда локально выгоднее направить движение.

Однако компас не выбирает за нас длину шага. Слишком длинный шаг может проскочить через подходящую область; слишком короткий будет продвигать нас медленно. Это уже вопрос алгоритма градиентного спуска — к нему мы подойдём в следующей главе.

Есть и ещё одна тонкость. «Самое быстрое направление» определяется относительно выбранных координат и единиц измерения. Если один параметр измеряется в тысячных долях, а другой — в десятках, их масштабы могут влиять на практическое поведение метода. Поэтому

параметры и признаки нередко нормируют: так компас работает в более разумно размеченной системе координат.

Наконец, если градиент равен нулю, стрелка замолкает. Это может быть минимум, но может быть максимум или седловая точка — место, где поверхность поднимается в одном направлении и опускается в другом. Нулевой градиент означает лишь отсутствие локального направления первого порядка, которое сразу указывает на изменение функции. Сам по себе он ещё не ставит печать «решение найдено».

Цепное правило: как передать чувствительность назад

На практике целевая функция редко записана одной короткой формулой. Чаще она складывается из последовательности простых вычислений: параметры создают предсказание, предсказание сравнивается с ответом, разница превращается в ошибку. Чтобы найти, как каждый исходный параметр влияет на конечный результат, нужно пройти эту цепочку связей. Здесь работает цепное правило.

Рассмотрим небольшую модель. Она получает вход (x) , умножает его на вес (w) , добавляет смещение (b) , возводит результат в квадрат и сравнивает с целевым значением (y) . Обозначим промежуточные вычисления:

Этап

Вычисление

Значение в примере

1

$$s=wx+b$$

$$(2)$$

2

$$a=s^2$$

$$(4)$$

3

$$r=a-y$$

$$(3)$$

4

$$L=\frac{1}{2}r^2$$

$$(4,5)$$

Возьмём $(x=2)$, $(w=1)$, $(b=0)$, $(y=1)$. Прямой проход прост: сначала получаем $(s=2)$, затем $(a=4)$, остаток $(r=3)$, а итоговая ошибка $(L=4,5)$.

Теперь спросим не только «какова ошибка?», но и «как она зависит от (w) и (b) ?». Начнём с последней операции. Производная $(\frac{1}{2}r^2)$ по (r) равна (r) , то есть здесь равна (3) . Это сообщает, насколько чувствительна ошибка к изменению остатка.

Остаток $(r=a-y)$ меняется вместе с (a) с коэффициентом (1) . Значит, чувствительность ошибки к (a) тоже равна (3) . Следующая операция — $(a=s^2)$. Её производная по (s) равна $(2s)$, то есть (4) . Умножаем локальные чувствительности: влияние (s) на итоговую ошибку равно $(3 \cdot 4 = 12)$.

Осталось вернуться к параметрам. Поскольку $(s=wx+b)$, его производная по (w) равна $(x=2)$, а по (b) — (1) . Поэтому:

$$\left[\frac{\partial L}{\partial w} = 12 \cdot 2 = 24, \quad \frac{\partial L}{\partial b} = 12 \cdot 1 = 12. \right]$$

Вектор градиента по параметрам $((w, b))$ равен $((24, 12))$. При текущем выборе параметров увеличение (w) или (b) локально повышает ошибку. Для её уменьшения нужно двигаться в противоположную сторону.

Цепное правило здесь выглядит как последовательная передача чувствительности от результата к источникам. Для функции, которая зависит от промежуточной величины, это записывается так:

$$\left[\frac{dL}{dw} = \frac{dL}{dr} \cdot \frac{dr}{da} \cdot \frac{da}{ds} \cdot \frac{ds}{dw}. \right]$$

Каждый множитель отвечает за один переход: как меняется следующий узел вычисления, если немного изменить предыдущий. Сложную производную мы не угадываем целиком — раскладываем на короткие локальные производные и перемножаем их. Именно поэтому градиенты можно эффективно считать даже для больших вычислительных цепочек: достаточно знать операции и связи между ними. Если один промежуточный результат влияет на несколько ветвей вычисления, вклады этих ветвей складываются.

Проверка: не верьте формуле вслепую

Даже аккуратное вычисление может содержать ошибку: пропущенный множитель, неверный знак, неправильный индекс. Полезно проверить производную численно — конечными разностями. Идея проста: слегка увеличиваем параметр, слегка уменьшаем его и смотрим, соответствует ли изменение функции нашей формуле.

Для параметра (w) используем центральную разность:

$$\left[\frac{\partial L}{\partial w} \approx \frac{L(w+h) - L(w-h)}{2h}. \right]$$

В нашем примере, если $(b=0)$, то $(L(w) = \frac{1}{2}(4w^2 - 1)^2)$. Аналитически мы получили $(\frac{\partial L}{\partial w} = 24)$ при $(w=1)$. Возьмём $(h=0.0001)$. Центральная разность даст приблизительно (24.00000032) — почти то же самое. Значит, формула, по крайней мере в этой точке, выдержала проверку.

Почему нельзя выбрать любое (h) ? Если шаг слишком велик, мы измеряем наклон на заметном участке кривой, а не локальную скорость в точке. Кривизна поверхности искажает результат. Если шаг слишком мал, значения $(L(w+h))$ и $(L(w-h))$ становятся почти одинаковыми. При вычитании теряется точность: вычислительная арифметика не хранит бесконечное число знаков, а разность близких чисел особенно чувствительна к округлению.

Практический подход — проверить производную несколькими разумными значениями (h) . Если результаты заметно расходятся, возможно, шаги слишком большие или слишком маленькие, функция шумная либо в точке есть излом. Центральная разность обычно точнее односторонней для гладкой функции, но требует двух вычислений вместо одного. Для параметров разного масштаба полезно подбирать и разные шаги проверки.

Вот короткий маршрут для самостоятельной проверки:

1. Выберите точку и параметр, например $(w=1)$, $(b=0)$.
2. Найдите аналитическое значение частной производной.
3. Вычислите конечную разность с небольшим шагом (h) .
4. Повторите проверку с шагом примерно в десять раз меньше и в десять раз больше.
5. Сравните результаты и проверьте, не объясняется ли расхождение округлением, кривизной или особенностью функции.

Численная проверка не заменяет математическое рассуждение. Она выполняет другую работу: ловит ошибки в формулах и программной реализации. В большом проекте это может быть разница между уверенным обучением модели и неделями поисков причины, по которой она не движется туда, куда должна.

Что взять с собой

Производная измеряет локальную скорость изменения функции. Частные производные показывают чувствительность к отдельным параметрам, а градиент объединяет эти чувствительности в вектор. Геометрически он перпендикулярен линии уровня и направлен к наиболее быстрому росту; чтобы уменьшать целевую функцию, нужно двигаться против него. Если вычисления состоят из цепочки операций, цепное правило позволяет передать влияние результата назад, шаг за шагом. А конечные разности дают простой способ проверить, верны ли полученные производные.

Теперь случайные изменения параметров можно заменить осмысленным вопросом: «Как функция отреагирует на такой шаг?» Следующий шаг — превратить ответ градиента в последовательность движений, которая будет систематически приближать нас к меньшей ошибке.

Градиентный спуск: от первой итерации к работающему алгоритму

В 23:17 разработчик смотрит на график прогноза и видит знакомую неприятность: модель вроде обучилась, но промахивается снова и снова. У неё всего два настраиваемых параметра — два условных регулятора, которыми можно поправить поведение. Но как именно оптимизатор решает, куда крутить каждый из них? Почему после одной итерации ошибка уменьшается, а после другой может взлететь?

Чтобы увидеть работу градиентного спуска изнутри, не нужны большая нейросеть и часы вычислений. Достаточно функции двух переменных, карандаша и нескольких аккуратных шагов. Представим, что $\theta(x)$ и $\theta(y)$ — отклонения параметров модели от их лучших значений. Её ошибку задаёт простая функция:

$$f(x, y) = x^2 + 2y^2.$$

Минимум здесь находится в точке $(0, 0)$: чем ближе параметры к ней, тем меньше ошибка. При этом отклонение по y обходится дороже, чем такое же отклонение по x : его вклад в ошибку вдвое больше. Это учебный пример, но механика, которую мы сейчас разберём, та же, что работает в более сложных моделях.

Одна итерация: посчитать, посмотреть, шагнуть

Начнём с параметров $(2, 1)$. Сначала вычислим ошибку:

$$f(2, 1) = 2^2 + 2 \cdot 1^2 = 6.$$

Затем найдём градиент — соберём частные производные по обоим переменным:

$$\nabla f(x, y) = \begin{pmatrix} 2x \\ 4y \end{pmatrix}.$$

В стартовой точке он равен

$$\nabla f(2, 1) = \begin{pmatrix} 4 \\ 4 \end{pmatrix}.$$

Градиент указывает направление наиболее быстрого роста функции. Значит, для уменьшения ошибки будем двигаться в противоположную сторону. Правило обновления параметров выглядит так:

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k),$$

где:

- θ_k — текущие параметры на шаге k ;
- $\nabla f(\theta_k)$ — градиент, вычисленный именно в текущей точке;
- η — скорость обучения, или размер шага;
- θ_{k+1} — новые параметры после обновления.

В нашем примере $\theta_0 = (2, 1)$. Возьмём $\eta = 0,1$. Тогда:

$$\theta_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} - 0,1 \begin{pmatrix} 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 1,6 \\ 0,6 \end{pmatrix}.$$

Ошибка в новой точке:

$$f(1,6, 0,6) = 1,6^2 + 2 \cdot 0,6^2 = 2,56 + 0,72 = 3,28.$$

Она снизилась с 6 до 3,28. Вот первая итерация во всей полноте: мы вычислили значение функции, нашли градиент, выбрали размер шага, обновили параметры и проверили результат.

Обратите внимание: градиент показывает, куда функция растёт быстрее всего, но шаг не обязан приводить сразу к минимуму. Он только предлагает направление и расстояние. Подобно тому как один поворот руля меняет курс машины, но ещё не доставляет её к месту назначения.

Несколько шагов на бумаге

Теперь повторим процедуру. На каждом шаге градиент надо вычислять заново: по мере изменения параметров меняется и направление спуска. В таблице указаны текущие координаты, градиент в этой точке, следующие координаты и ошибка после перехода.

Шаг k

Текущие параметры (x, y)

Градиент ∇f

Новые параметры (x, y)

Ошибка в новой точке

0

$(2, 1)$

$(4, 4)$

$(1, 6, 0, 6)$

$(3, 28)$

1

$(1, 6, 0, 6)$

$(3, 2, 2, 4)$

$(1, 28, 0, 36)$

$(1, 8976)$

2

$(1, 28, 0, 36)$

$(2, 56, 1, 44)$

$(1, 1024, 0, 216)$

$(1, 141888)$

3

$(1, 1024, 0, 216)$

$(2, 1048, 0, 864)$

$(0, 8192, 0, 1296)$

$(0, 704517)$

Проверим второй шаг. В точке $(1, 6, 0, 6)$ градиент равен $(3, 2, 2, 4)$. Поэтому:

$$\nabla f(1, 6, 0, 6) - 0 = 1(3, 2, 2, 4) = (1, 28, 0, 36).$$

Ошибка там равна $1^2 + 28^2 + 0^2 + 36^2 = 1, 8976$. Всё повторяется по одному и тому же правилу, а параметры постепенно приближаются к нулю.

Эта таблица полезна не только как иллюстрация. Она позволяет отлаживать алгоритм без доверия к «чёрному ящику»: если программа на тех же исходных условиях выдаёт совершенно другие координаты, можно проверить вычисления по шагам. Например, первая новая точка обязана быть $(1, 6, 0, 6)$. Если это не так, ошибка возникла раньше, чем алгоритм успел стать сложным.

Ловушка, в которую легко попасть

В какой-то момент разработчик замечает: производные вычисляются правильно, но ошибка после обновления почему-то растёт. Причина оказывается в одной мелочи — знаке.

Если вместо вычитания прибавить градиент, получится:

$$\nabla f = \begin{pmatrix} 2 \\ 1 \end{pmatrix} + 0, 1 \begin{pmatrix} 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 2, 4 \\ 1, 4 \end{pmatrix}.$$

Тогда ошибка станет

$$\nabla f(2, 4, 1, 4) = 2^2 + 2 \cdot 1^2 + 4^2 + 4^2 = 9, 68,$$

то есть вырастет с 6 до 9,68. Формула градиента записана верно, вычисление градиента верно, размер шага тот же — но программа пошла вверх вместо того, чтобы спускаться.

Правило простое: для минимизации вычитаем градиент. Но полезнее, чем механически заучить знак, — проверять результат. На учебном примере после разумного шага ошибка должна уменьшиться. Если она растёт, проверьте направление обновления и размер шага. Один такой тест часто быстрее находит проблему, чем долгий поиск ошибки в остальном коде.

Есть и другая тонкость: все компоненты градиента нужно считать в одной и той же текущей точке. Сначала вычислите полный градиент, затем одновременно обновите параметры. Если изменить $\mathbf{x}$, а потом пересчитать производную по y уже с новым $\mathbf{x}$, получится другой алгоритм — последовательное покоординатное обновление. Оно тоже может быть полезным, но это уже не та итерация, которую мы записали в формуле.

От ручного счёта к алгоритму

Когда процесс повторяется, его можно изложить словами:

1. Скопировать начальные параметры, чтобы случайно не изменить исходный объект.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.