Градиентный спуск

Веса можно подбирать вручную, глядя на карту. А как программа сама решает, в какую сторону и насколько их сдвинуть?

В ручном опыте мы меняли веса, смотрели на границу и сравнивали среднюю ошибку. Теперь поручим этот поиск программе. Карта, координаты и правильные метки остаются прежними; меняются **параметры модели**. Для одного нейрона это два веса и смещение, для сети со скрытыми слоями — много таких чисел.

Начнём с двух облаков. Модель уже умеет проводить подходящую прямую, поэтому можно сосредоточиться на обучении. Нам предстоит понять, как алгоритм выбирает направление изменения и почему более крупное обновление не всегда полезнее.

1. Как понять, куда менять вес

Зафиксируем обучающие точки и все параметры, кроме первого веса. Мысленно чуть увеличим этот вес и пересчитаем среднюю ошибку. Если она растёт, полезное локальное направление — уменьшать вес. Если падает — увеличивать. Это та же проверка, которую мы делали ползунком, но программе нужна численная подсказка.

Градиент содержит для каждого параметра такую подсказку: насколько и в какую сторону изменяется ошибка при небольшом изменении этого параметра. Он относится к текущим весам и текущим примерам. После обновления подсказки нужно вычислить заново.

Градиентный спуск меняет параметры в противоположную сторону — против локального роста ошибки. Если для первого веса подсказка положительная, вес уменьшают; если отрицательная — увеличивают. Остальные веса могут в тот же момент двигаться в другие стороны.

Это не движение робота по местности. Точки карты остаются на своих координатах. Меняются числа внутри нейрона, а значит, меняется его ответ сразу во многих местах и движется цветная граница. Не стоит также путать градиент ошибки по весам с направлением на север или юг на карте.

> При текущей настройке небольшое увеличение первого веса повышает ошибку на обучающих точках. В какую сторону направлен локальный шаг спуска по этому весу?
>
> ○ В сторону уменьшения веса
>
> ○ К большему номеру примера
>
> ○ Всегда к нулю
>
> ○ Всегда в сторону увеличения

Ранее

Функция потерь уже измеряет ошибку на точках карты робота. Теперь разберём, как по ней автоматически менять веса и смещение.

← Обучение нейронных сетей: функция потерь

2. Насколько большим будет изменение

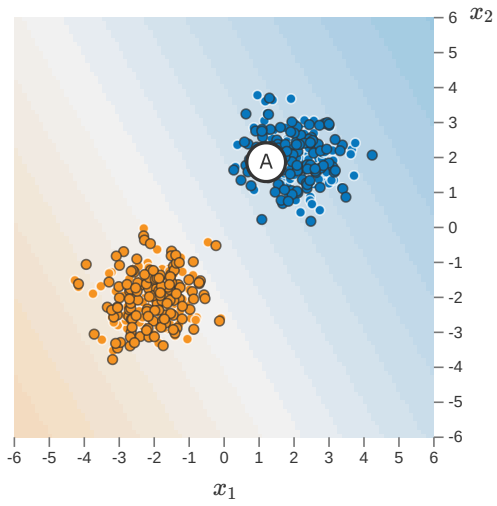
Скорость обучения η — коэффициент, которым масштабируют градиент. Для одного веса обновление можно записать так:

$w_{\text{новый}} = w_{\text{старый}} - \eta g.$

Здесь g — подсказка градиента для этого веса. Например, при $w = 0{,}5$, $g = -0{,}4$ и $\eta = 0{,}25$ новый вес равен $0{,}6$: вычитание отрицательного числа увеличило вес. Если величина градиента станет меньше, при той же скорости изменение тоже уменьшится. Поэтому скорость обучения не означает фиксированное расстояние каждого шага.

Попробуйте. В тренажёре оставьте «Два облака», выполните несколько отдельных проходов кнопкой ▶|. После каждого смотрите на веса, смещение, ошибку выбранной точки и среднюю ошибку. Затем запустите непрерывное обучение и остановите его, когда граница разделит облака.

Алгоритм меняет веса на знакомой карте



Два облака. Синий класс — безопасно, оранжевый — опасно.

Состояние опыта: 0 проходов. Скрытых слоёв нет.

Веса: 0,056; 0,032. Смещение: 0,1.

Точка A: (1,203; 1,866). Известный класс: безопасно. Прогноз: безопасно.

$a \approx 0{,}223,\ y = 1,\ \ell = \frac{(a-y)^2}{2} \approx 0{,}302$

Веса: 0,056; 0,032. Смещение: 0,1. Ошибка на обучении: 0,343; на проверке: 0,352.

Теперь открыта скорость обучения. Один проход включает несколько обновлений по порциям данных. Точки A/B/C помогают проследить конкретные ответы.

Интерактивный эксперимент — в электронной версии урока.

Сначала запишите начальные веса и ошибки. Для сравнения скоростей нажмите сброс, убедитесь, что исходные веса те же, измените только скорость обучения и выполните такое же число проходов. Например, сравните $0{,}001$, $0{,}03$ и 1. Координаты и метки не меняйте: иначе будет трудно понять причину различий.

Малый коэффициент обычно даёт меньшие обновления, но результат зависит и от величины градиента. Большой коэффициент может быстрее уменьшить ошибку или перескочить полезную область. На простых облаках даже большой шаг иногда работает хорошо — не нужно искать обязательную поломку. Вывод делайте по фактическим параметрам и обеим ошибкам в своём запуске.

> Вес был $0{,}5$, после обновления стал $0{,}6$. На сколько он увеличился за это обновление?

Заметка

Один проход по данным и одно обновление — разные действия. В этом опыте внутри прохода веса несколько раз меняются по порциям из десяти примеров.

3. Один цикл обучения

Сначала сеть получает координаты обучающих точек и вычисляет выходы. Это **прямой проход**: от входов через нейроны к ответу. Затем функция потерь сравнивает выходы с известными метками $+1$ и -1.

Чтобы узнать влияние каждого веса на ошибку, используют **обратное распространение ошибки**. Метод проходит цепочку вычислений в обратном порядке: от ошибки к выходу, затем к более ранним нейронам и их параметрам. Он вычисляет градиенты. Само изменение весов выполняет алгоритм оптимизации, например градиентный спуск.

Координаты x_1, x_2	→	Ответ сети a	→	Сравнение с меткой $\ell = (a - y)^2/2$	→	Градиенты по весам	→	Обновление параметров

После обновления подаём координаты снова: сеть вычислит ответы уже с новыми весами. Известные метки остаются прежними.

В нашем Playground алгоритм берёт порцию из десяти обучающих точек, вычисляет средний градиент для неё и обновляет параметры. Затем обрабатывает следующую порцию. Один проход на счётчике означает обход всего обучающего набора, а не одно изменение веса. Поэтому после нажатия ►| вы видите результат нескольких обновлений.

График ошибки по проходам тоже имеет свой смысл: по горизонтали — ход обучения, по вертикали — ошибка текущей модели. Это не сама карта и не график ошибки по одному весу. На карте оси по-прежнему обозначают координаты места.

Ошибка выбранной точки может вырасти после полезного для среднего обновления. Кроме того, соседние порции различаются: шаг в пользу одной порции не обязан улучшать все остальные. Смотрите на общую динамику и проверочный набор, а не только на одно значение.

> После увеличения скорости обучения ошибка заметно колеблется и не улучшается. Что разумно проверить первым?
>
> - ○ Изменить правильные метки точек
> - ○ Скрыть растущие значения ошибки
> - ○ Объявить любой проход улучшением
> - ○ Уменьшить скорость обучения и повторить опыт с того же старта

Заметка

Для каждого места входами остаются две координаты. Во время обучения меняются общие параметры модели, а известные координаты и метки сохраняются.

4. Чего обучение не меняет

Выберите «Кольцо» в том же тренажёре. Модель всё ещё состоит из одного нейрона. Попробуйте несколько скоростей и сравните результат с облаками. Даже удачный подбор параметров не позволит прямой границе окружить центральный класс.

Мы уже решали это ограничение в сюжете о слоях: добавляли несколько нейронов с нелинейной активацией. Градиентный метод остаётся тем же, но теперь он подбирает больше параметров в более выразительной модели. Обратное распространение учитывает влияние раннего слоя через последующие слои.

Поэтому различайте два вопроса: умеет ли выбранная модель представить нужную границу и смог ли алгоритм найти хорошие параметры? Увеличение времени обучения или скорости не добавляет скрытые нейроны. А добавление нейронов само по себе не гарантирует хорошей настройки и переноса на новые места.

Обучение останавливают не только при нулевой ошибке. Данные могут содержать шум, а дальнейшее уменьшение ошибки на известных местах иногда ухудшает проверочный результат. Для сравнения настроек нужен одинаковый бюджет проходов и отдельная оценка качества.

После остановки робот может оценить новую пару координат. Выполняется только прямой проход с готовыми весами. Метка нового места не нужна для получения прогноза; она понадобится позже, чтобы проверить ответ. Одно применение модели не обучает её автоматически.

Продолжение

Мы проследили весь цикл на одной карте: координаты → ответ → ошибка → изменение параметров. Далее разберём, какие вычисления требуют ресурсов, а в общей лаборатории сравним несколько архитектур на этих же картах.

Градиент подсказывает локальное направление роста ошибки по каждому параметру; спуск идёт против него.

Скорость обучения масштабирует обновление; один проход по данным содержит несколько обновлений.

Обучение подбирает параметры внутри выбранной архитектуры, а качество проверяют на новых данных.

Домашняя работа

1. На двух облаках сравните две скорости обучения за одинаковые десять проходов с одного старта. Запишите начальные веса, обе скорости и ошибки на обучении и проверке. Объясните, какой вывод позволяет сделать этот опыт и чего он ещё не доказывает.

2. > Вес нейрона равен 2, алгоритм увеличивает его на 0,5. Каким станет вес?

Источники

- Оригинальный тренажёр взят из TensorFlow Playground (https://playground.tensorflow.org/). Авторы: Daniel Smilkov и Shan Carter.
- Прямой и обратный проход (https://education.yandex.ru/handbook/ml/article/pervoe-znakomstvo-s-polnosvyaznymi-nejrosetyami)
- Градиентный спуск (https://d2l.ai/chapter_optimization/gd.html)